altmult_add Megafunction

User Guide

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com

Quartus II Software Version: 7.0
Document Version: 2.3
Document Date: March 2007

Copyright © 2007 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

I.S. EN ISO 9001

UG-030805-2.3

Contents

About this User Guide v
- Revision History v
- How to Contact Altera v
- Typographic Conventions vi

Chapter 1. About this Megafunction
- Device Family Support 1–1
- Introduction 1–1
- Features 1–1
- General Description 1–1
  - Common Applications 1–4
- Resource Utilization and Performance 1–5

Chapter 2. Getting Started
- Software and System Requirements 2–1
- MegaWizard Plug-In Manager Customization 2–1
- MegaWizard Page Descriptions 2–1
  - Inferring Megafunctions from HDL Code 2–8
- Instantiating Megafunctions in HDL Code 2–9
- Identifying a Megafunction after Compilation 2–9
- Simulation 2–10
  - Quartus II Simulator 2–10
  - EDA Simulation 2–10
- SignalTap II Embedded Logic Analyzer 2–11
- Design Example: Complex Multiplication 2–11
  - Design Files 2–11
- Example 1 2–11
  - Generate a 9 × 9-Bit Multiplier/Adder (the Real Expression) 2–12
  - Generate a 9 x 9-Bit Multiplier/Adder (the Imaginary Expression) 2–21
  - Combine real_mult and image_mult to Create a Complex Multiplier 2–27
  - Implement the 9 × 9 Complex Multiplier Design 2–29
  - Functional Results—Simulate the Complex Multiplier Design 2–31
  - Timing Results 2–33
  - Simulate the Complex Multiplier Design in ModelSim-Altera 2–35
- Design Example: Implementing a Simple FIR Filter 2–37
  - Design Files 2–38
- Example 2 2–38
  - Generate a 4-Tap FIR Filter 2–38
  - Implement the FIR 4-Tap Design 2–48
  - Functional Results—Simulate the FIR 4-Tap Design 2–50
  - Timing Results 2–52

Simulate the FIR Filter Design in the ModelSim-Altera Software 2–54
Conclusion 2–56

Chapter 3. Specifications

Ports and Parameters 3–1

About this User Guide

Revision History

The table below displays the revision history for the chapters in this User Guide.

Date	Version	Changes Made
March 2007	2.3	• Added Cyclone III information to tables. No new screenshots were taken.
November 2006	2.2	• Made some minor adjustments pertaining to the Stratix III release.
November 2006	2.1	• Updated for the Quartus II software version 6.1 GUI changes and functionality. • Updated design examples.
August 2006	2.0	• Updated for the Quartus II software version 6.0 GUI changes and functionality. • Updated design examples.
March 2005	1.0	Initial release.

How to Contact Altera

For the most up-to-date information about Altera® products, go to the Altera world-wide web site at www.altera.com. For technical support on this product, go to www.altera.com/mysupport. For additional information about Altera products, consult the sources shown below.

Information Type	Contact
Technical support	www.altera.com/mysupport/
Product literature	www.altera.com
Altera literature services	literature@altera.com *(1)*
FTP site	ftp.altera.com

Note to table:
(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

This document uses the typographic conventions shown below.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: **Save As** dialog box.
bold type	External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: $\mathbf{f_{MAX}}$, **\qdesigns** directory, **d:** drive, **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Document titles are shown in italic type with initial capital letters. Example: *AN 75: High-Speed Board Design.*
Italic type	Internal timing parameters and variables are shown in italic type. Examples: t_{PIA}, $n + 1$. Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: *<file name>*, *<project name>***.pof** file.
Initial Capital Letters	Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.
"Subheading Title"	References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: "Typographic Conventions."
`Courier type`	Signal and port names are shown in lowercase Courier type. Examples: `data1`, `tdi`, `input`. Active-low signals are denoted by suffix `n`, e.g., `resetn`. Anything that must be typed exactly as it appears is shown in Courier type. For example: `c:\qdesigns\tutorial\chiptrip.gdf`. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword `SUBDESIGN`), as well as logic function names (e.g., `TRI`) are shown in Courier.
1., 2., 3., and a., b., c., etc.	Numbered steps are used in a list of items when the sequence of the Items is important, such as the steps listed in a procedure.
■ ● •	Bullets are used in a list of items when the sequence of the items is not important.
✓	The checkmark indicates a procedure that consists of one step only.
	The hand points to information that requires special attention.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or the user's work.
	A warning calls attention to a condition or possible situation that can cause injury to the user.
↵	The angled arrow indicates you should press the Enter key.
	The feet direct you to more information on a particular topic.

1. About this Megafunction

Device Family Support

The altmult_add megafunction supports the following target Altera® device families:

- Stratix® series
- Cyclone® series
- HardCopy® series
- MAX® series
- APEX™ series
- ACEX® devices
- FLEX® series

Introduction

As design complexities increase, use of vendor-specific IP blocks has become a common design methodology. Altera provides parameterizable megafunctions that are optimized for Altera device architectures. Using megafunctions instead of coding your own logic saves valuable design time. Additionally, the Altera-provided functions may offer more efficient logic synthesis and device implementation. You can scale the megafunction's size by simply setting parameters.

Features

The altmult_add megafunction implements a basic adder/multiplier and offers many additional features including:

- Parameterizable input data and output data widths
- Active high asynchronous clear and clock-enable control inputs
- Support for both signed and unsigned data representation formats
- Ability to add or subtract the product pair
- Facility to added extra latency to have additional registers required to pipeline the output of the multiplier/adder

General Description

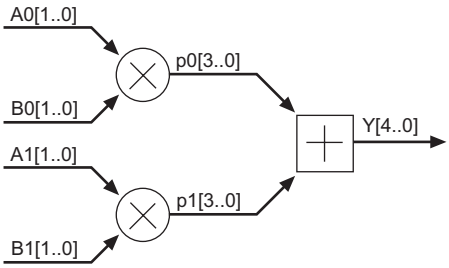
The altmult_add megafunction allows you to implement a multiplier/adder. A multiplier/adder accepts pairs of inputs. The members of each pair are multiplied together and the multiplier/adder then adds or subtracts the products of all other pairs. This function can be expressed as an equation (Equation 1).

$$y = A_0B_0 \pm A_1B_1 \pm .. \pm A_nB_n \quad (1)$$

In this equation, *n* denotes the number of pairs.

Figure 1–1 illustrates a basic multiplier/adder, with two pair of 2-bit inputs.

Figure 1–1. Basic 2-bit Input Multiplier/Adder

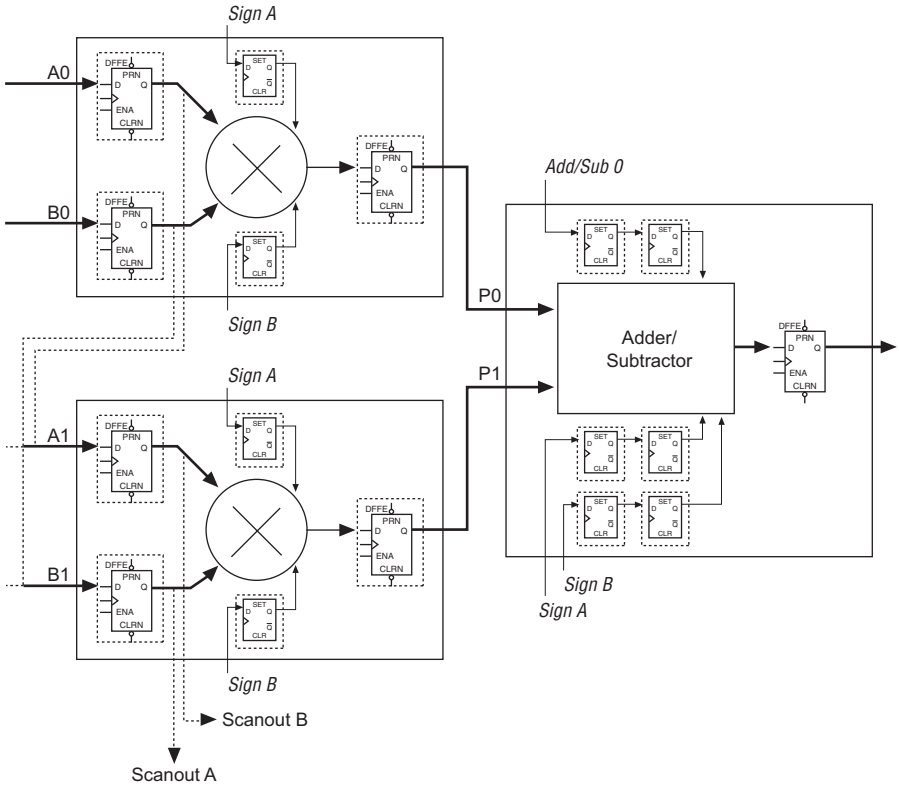


The altmult_add variations provide the following options for your design. These include the ability to:

- Change the operation to subtraction instead of addition
- Accommodate the signed/unsigned data representation
- Provide data shifting chains and additional registers
- Control asynchronous clear of the registers
- Use a DSP block in the multiplier/adder
- Size the data width of the input and output

A multiplier/adder with these optional features is shown in Figure 1–2.

Figure 1–2. Multiplier/Adder with Optional Features



The altmult_add megafunction implements the multiplier/adder described in Figure 1–2, and offers many variations in dedicated digital signal processing (DSP) block circuitry. Data input sizes of up to 18-bits are accepted. Because the DSP blocks allow for one or two levels of 2-input add/subtract operations on the product, this function creates up to four multipliers. Additional features include dynamically changing signed or unsigned data support, dynamically changing add/subtract-based operation, setting up data shifting chains and additional registers for latency, and controlling asynchronous clear of the registers.

The multipliers and adders of the altmult_add megafunction are placed in the dedicated DSP Block circuitry of Stratix devices. If all of the input data widths are 9 bits wide or smaller, the function uses the 9 × 9-bit input multiplier configuration in the DSP Block. If not, the DSP block uses 18 × 18-bit input multipliers to process data with widths between 10 bits

and 18 bits. If multiple altmult_add megafunctions occur in a design, the functions are distributed to as many different DSP blocks as possible so that routing to these blocks is more flexible. Fewer multipliers per DSP block allow more routing choices into the block by minimizing paths to the rest of the device.

The registers and extra pipeline registers for the following signals are also placed inside the DSP block:

- Data input
- Signed/unsigned select
- Add/subtract select
- Products of multipliers

In the case of the output result, the first register is placed in the DSP block, however the extra latency registers are placed in logic elements outside the block. Peripheral to the DSP block, including data inputs to the multiplier, control signal inputs, and outputs of the adder, use regular routing to communicate with the rest of the device. All connections in the function use dedicated routing inside the DSP block. This dedicated routing includes the shift register chains when you select the option to shift a multiplier's registered input data from one multiplier to an adjacent multiplier.

Common Applications

Multiplier/adder applications include serial finite impulse response (FIR) filters with fixed or variable coefficients, fast fourier transform (FFT), and additional designs requiring a serial summation of products. Designing with the altmult_add megafunction is most efficient when a design requires the summation of products. Such designs can take advantage of the speed offered by the dedicated DSP circuitry blocks.

Use the altmult_add megafunction with devices that offer DSP blocks. Currently this feature is available in Stratix devices. When accumulating results from multipliers in the Stratix devices, consider using the altmult_accum megafunction as described in the *altmult_accum Megafunction User Guide*. If it is required to place adding or subtracting results of multipliers in logic elements or in device families other than Stratix devices, consider using lpm_mult and lpm_add_sub megafunctions together.

For details about these megafunctions, refer to the *lpm_mult Megafunction User Guide*, the *lpm_add_sub Megafunction User Guide*, and the *altmult_accum Megafunction User Guide*.

For more information about the multiply/add mode of the Stratix series DSP blocks and details on using accumulators in FIR filter applications, refer to the *Stratix II Architecture* chapter in volume 1 of the *Stratix II Handbook*, *DSP Blocks in Stratix and Stratix GX Devices*, and *Implementing High-Performance DSP Functions in Stratix and Stratix GX Devices* chapters in volume 2 of the *Stratix Device Handbook*.

Resource Utilization and Performance

The altmult_add megafunction is implemented using either logic resources or dedicated multiplier circuitry in Altera devices. Typically, the altmult_add megafunction is translated to the dedicated multiplier circuitry when available, providing improved performance and resource utilization.

For detailed information about the architecture of the DSP blocks and embedded multipliers, and detailed information on the hardware conversion process, refer to the following sources:

- *Introduction* and *Stratix Architecture* chapters in volume 1 of the *Stratix Device Handbook*, *DSP Blocks in Stratix and Stratix GX Devices*, and *Implementing High-Performance DSP Functions in Stratix and Stratix GX Devices* chapters in volume 2 of the *Stratix Device Handbook*
- *Introduction* and *Stratix II Architecture* chapters in volume 1 of the *Stratix II Device Handbook*, and *DSP Blocks in Stratix II Devices* and *Configuring Stratix II Devices* chapters in volume 2 of the *Stratix II Device Handbook*
- *Introduction* and *Cyclone II Architecture* chapters in volume 1 of the *Cyclone II Device Handbook*, and *Embedded Multipliers in Cyclone II Devices* and *Configuring Cyclone II Devices* chapters in volume 2 of the *Cyclone II Device Handbook*

Table 1–1 summarizes the resource usage for an altmult_add function used to implement an 8-bit signed multiplier.You can force the compiler to implement the function in logic resources or DSP blocks, or allow the compiler to use the default implementation.

Table 1–1. altmult_add Resource Usage (Part 1 of 2) Note (1)

Device Family	Optimization (2)	Width	Logic Elements	DSP Blocks
Stratix II	Balanced	9-bit	—	2
	Balanced	16-bit	—	4
	Balanced	9-bit	182 ALUT	—
	Balanced	16-bit	513 ALUT	—

Table 1–1. altmult_add Resource Usage (Part 2 of 2) Note (1)

Device Family	Optimization (2)	Width	Logic Elements	DSP Blocks
Stratix GX	Balanced	9-bit	—	—
	Balanced	16-bit	—	—
	Balanced	9-bit	—	—
	Balanced	16-bit	—	—
Stratix	Balanced	9-bit	—	2
	Balanced	16-bit	—	4
	Balanced	9-bit	311	—
	Balanced	16-bit	777	—
HardCopy Stratix	Balanced	9-bit	—	2
	Balanced	16-bit	—	4
	Balanced	9-bit	324	—
	Balanced	16-bit	798	—
Cyclone II	Balanced	9-bit	293	—
	Balanced	16-bit	756	—
Cyclone	Balanced	9-bit	311	—
	Balanced	16-bit	777	—

Note to Table 1–1:

(1) The performance information is available from the MegaWizard. The information in this table is valid and accurate as of this document release.

(2) Choose a design implementation that balances high performance with minimal logic usage. This setting is available for APEX 20K, Cyclone series, MAX II, Stratix, and Stratix II devices only. The balanced optimization logic option can be set in **Analysis and Synthesis settings** (Assignments menu).

The MegaWizard® Plug-In Manager reports approximate resource utilization based on user specification and parameters, available in the lower left corner of the MegaWizard Plug-In Manager screen.

2. Getting Started

Software and System Requirements

The instructions in this section require the following hardware and software:

- For operating system support information, refer to the support page of the Altera website (**www.altera.com**)
- The Quartus® II software version 6.1 or later

MegaWizard Plug-In Manager Customization

Use the MegaWizard® Plug-In Manager to specify the altmult_add megafunction features for each multiplier in your design.

In the Quartus II software, start the MegaWizard Plug-In Manager in one of the following ways:

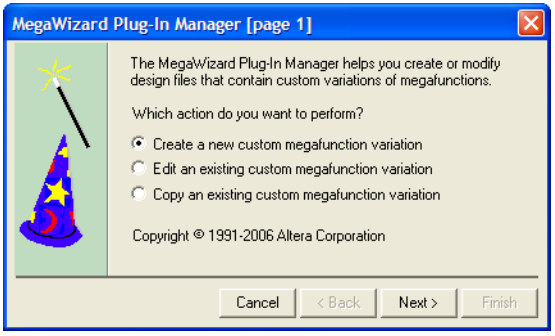
- On the Tools menu, click **MegaWizard Plug-In Manager**.
- Double-click in the Block Editor to open the **Symbol** dialog box. Click **MegaWizard Plug-In Manager**.
- Start the stand-alone version of the MegaWizard Plug-In Manager by typing the following command at the command prompt:
  `qmegawiz↵`

MegaWizard Page Descriptions

This section provides descriptions of the options available on the individual pages of the altmult_add wizard.

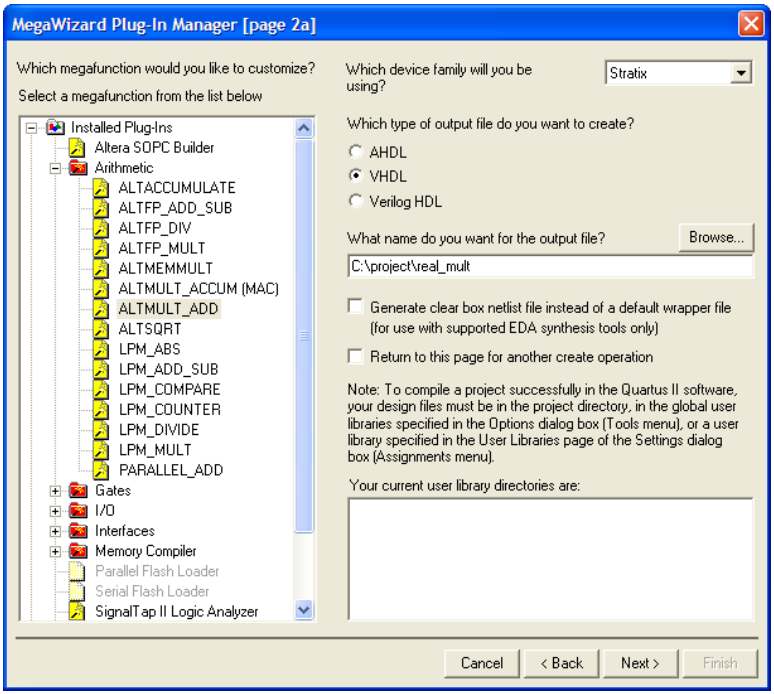
Page 1 of the MegaWizard Plug-In Manager is shown in Figure 2–1.

Figure 2–1. MegaWizard Plug-In Manager [Page 1]


MegaWizard Plug-In Manager [page 1]
The MegaWizard Plug-In Manager helps you create or modify design files that contain custom variations of megafunctions.
Which action do you want to perform?
Create a new custom megafunction variation
Edit an existing custom megafunction variation
Copy an existing custom megafunction variation
Copyright © 1991-2006 Altera Corporation
Cancel
< Back
Next >
Finish

You can choose to create, edit, or copy a custom megafunction variation. On page 2a of the altmult_add wizard, specify the plug-in, family of device you want to use, type of output file to create, and the name of the output file (Figure 2–2). Choose AHDL (**.tdf**), VHDL (**.vhd**), or Verilog HDL (**.v**) as the output file type. You can also create a clearbox instantiation for third-party EDA tools.

Figure 2–2. MegaWizard Plug-In Manager [Page 2a]


MegaWizard Plug-In Manager [page 2a]
Which megafunction would you like to customize?
Select a megafunction from the list below
Installed Plug-Ins
Altera SOPC Builder
Arithmetic
ALTACCUMULATE
ALTFP_ADD_SUB
ALTFP_DIV
ALTFP_MULT
ALTMEMMULT
ALTMULT_ACCUM (MAC)
ALTMULT_ADD
ALTSQRT
LPM_ABS
LPM_ADD_SUB
LPM_COMPARE
LPM_COUNTER
LPM_DIVIDE
LPM_MULT
PARALLEL_ADD
Gates
I/O
Interfaces
Memory Compiler
Parallel Flash Loader
Serial Flash Loader
SignalTap II Logic Analyzer
Which device family will you be using?
Stratix
Which type of output file do you want to create?
AHDL
VHDL
Verilog HDL
What name do you want for the output file?
Browse...
C:\project\real_mult
Generate clear box netlist file instead of a default wrapper file (for use with supported EDA synthesis tools only)
Return to this page for another create operation
Note: To compile a project successfully in the Quartus II software, your design files must be in the project directory, in the global user libraries specified in the Options dialog box (Tools menu), or a user library specified in the User Libraries page of the Settings dialog box (Assignments menu).
Your current user library directories are:
Cancel
< Back
Next >
Finish

On page 3 of the altmult_add wizard, specify the number of multipliers to be created, the width of the input bus, whether the input bus is signed, unsigned, or variable. You can also set options to configure `signa` and `signb` inputs (Figure 2–3).

Figure 2–3. MegaWizard Plug-In Manager—altmult_add [Page 3 of 7]

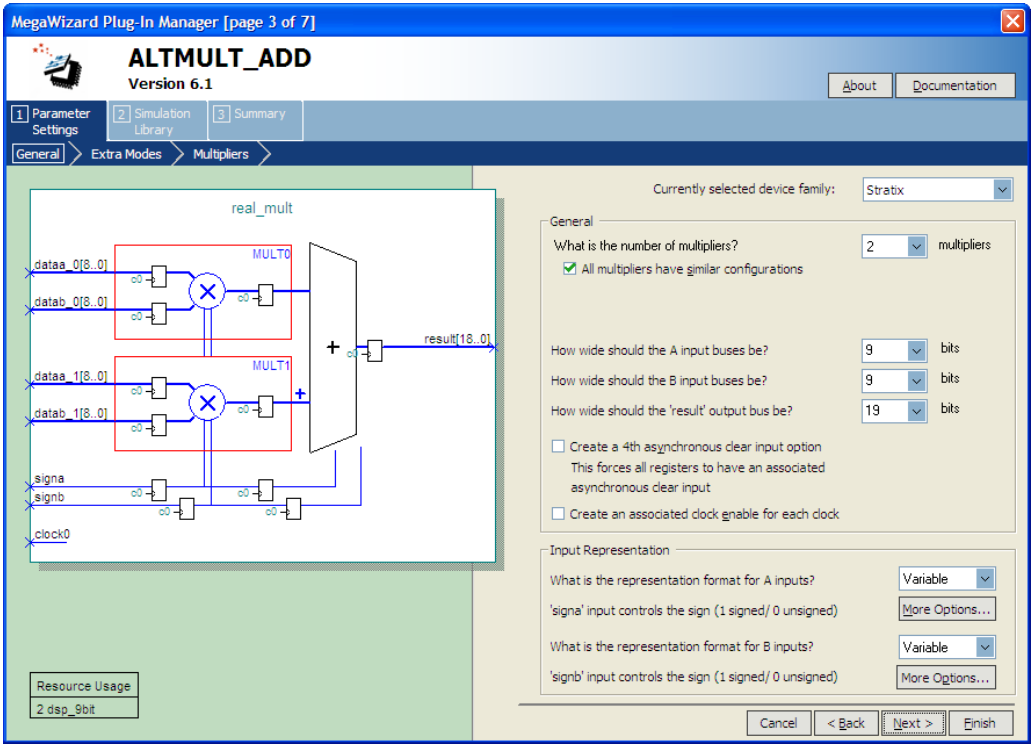


Table 2–1 shows the options available on page 3 of the altmult_add MegaWizard Plug-In Manager.

Table 2–1. altmult_add Plug-In Manager (Page 3) Options (Part 1 of 2)

Function	Description
Currently selected device family	Specify the family device you want to use.
What is the number of multipliers?	Specify the number of multipliers.
All multipliers have similar configurations	Specify if all the multipliers have the same configurations. If unchecked, individual settings is done for every multiplier. This option is selected by default when device is Stratix® III.
Add hardware support for hardware saturation and rounding. This will force all inputs to be in Q1.15 format.	Available when device is Cyclone II, HardCopy II, Stratix II or Stratix II GX. Select hardware saturation and rounding support. Default rounding for all input is set to Q1.15 format (fixed point arithmetic notation with 15 bits of precision).

Table 2–1. altmult_add Plug-In Manager (Page 3) Options (Part 2 of 2)

Function	Description
Add hardware support for hardware saturation and rounding.	Only available when device is Stratix III. Select hardware saturation and rounding support.
How wide should the A input buses be?	Specify the width of A input buses. If device is Cyclone II, HardCopy II, Stratix II, Stratix II GX or Stratix III, the width of A is fixed at 16 bits when hardware saturation and rounding is selected.
How wide should the B input buses be?	Specify the width of B input buses. If device is Cyclone II, HardCopy II, Stratix II, Stratix II GX or Stratix III, the width of A is fixed at 16 bits when hardware saturation and rounding is selected.
How wide should the 'result' output bus be?	Specify the width of 'result' output buses.
In what format should the 'result' output bus be?	Available when device is Cyclone II, HardCopy II, Stratix II, Stratix II GX or Stratix III and if the hardware saturation and rounding option is selected. Specify the rounding format of 'result' output bus.
Create a 4th asynchronous clear input option. This forces all registers to have an associated asynchronous clear input.	Select to create an asynchronous clear input.
Create an associated clock enable for each clock.	Select to create an associated clock enable for each clock.
What is the representation format for A inputs?	Specify the representation format for A inputs. When device is HardCopy™ II, Stratix II, Stratix II GX or Stratix III and if the hardware saturation and rounding option is selected, representation format for A inputs is default to 'signed'.
What is the representation format for B inputs?	Specify the representation format for B inputs. When device is HardCopy II, Stratix II, Stratix II GX or Stratix III and if the hardware saturation and rounding option is selected, representation format for B inputs is default to 'signed'.
'signa' input controls the sign (1 signed/0 unsigned).	Available when representation format for A inputs is 'variable'. Specify the input register and/or pipeline register.
'signb' input controls the sign (1 signed/0 unsigned)	Available when representation format for B inputs is 'variable'. Specify the input register and/or pipeline register.

On page 4 of the altmult_add wizard, specify which operation is performed on the first pair of Add/Sub multipliers and select the implementation type. Specify whether to use default or dedicated multiplier circuitry, or logic elements (Figure 2–4).

This parameter is available only for Stratix series and HardCopy Stratix devices.

Figure 2–4. MegaWizard Plug-In Manager —altmult_add [Page 4 of 7]

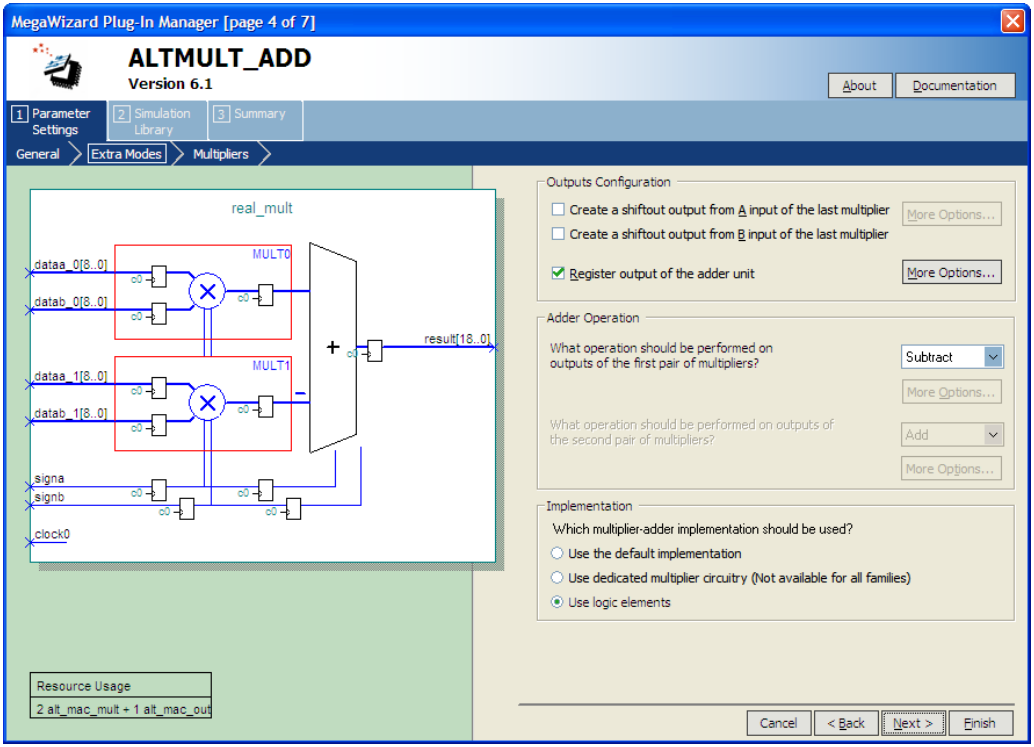


Table 2–2 shows the options available on page 4 of the altmult_add MegaWizard Plug-In Manager.

Table 2–2. altmult_add Plug-In Manager (Page 4) Options (Part 1 of 2)

Function	Description
Create a shiftout output from A input of the last multiplier	Select to create a shiftout output. When device is Stratix III, **More Options** is available for shiftout register configuration.
Create a shiftout output from B input of the last multiplier	Select to create a shiftout output. This option is not available when device is Stratix III.
Register output of the adder unit	Select to create register to the output.
What operation should be performed on outputs of the first pair of multipliers?	Specify the operation to be performed on the outputs of the first pair of multipliers.
What operation should be performed on outputs of the second pair of multipliers?	Specify the operation to be performed on the outputs of the second pair of multipliers. Note: Only available when the number of multipliers is 4.

Table 2–2. altmult_add Plug-In Manager (Page 4) Options (Part 2 of 2)	
Function	**Description**
Implementation	Specify implementation configurations.
Which multiplier-adder implementation should be used?	Specify which implementation to use for the multiplier-adder. 'Use dedicated multiplier circuitry' is only available on devices which have DSP block support.

On page 5 of the altmult_add megafunction wizard, specify where the inputs of the multiplier are connected. Connect inputs to either the multiplier input or shift input. Register the multiplier outputs and inputs (Figure 2–5).

Figure 2–5. MegaWizard Plug-In Manager —altmult_add [Page 5 of 7]

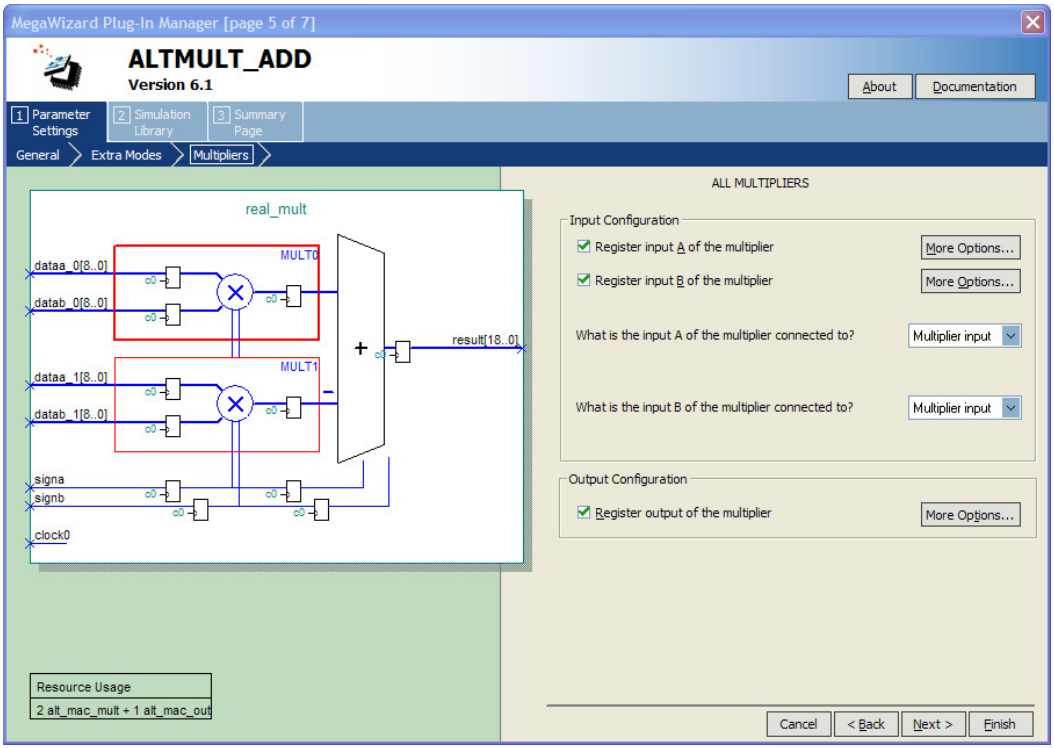


Table 2–3 shows the options available on page 5 of the altmult_add MegaWizard Plug-In Manager.

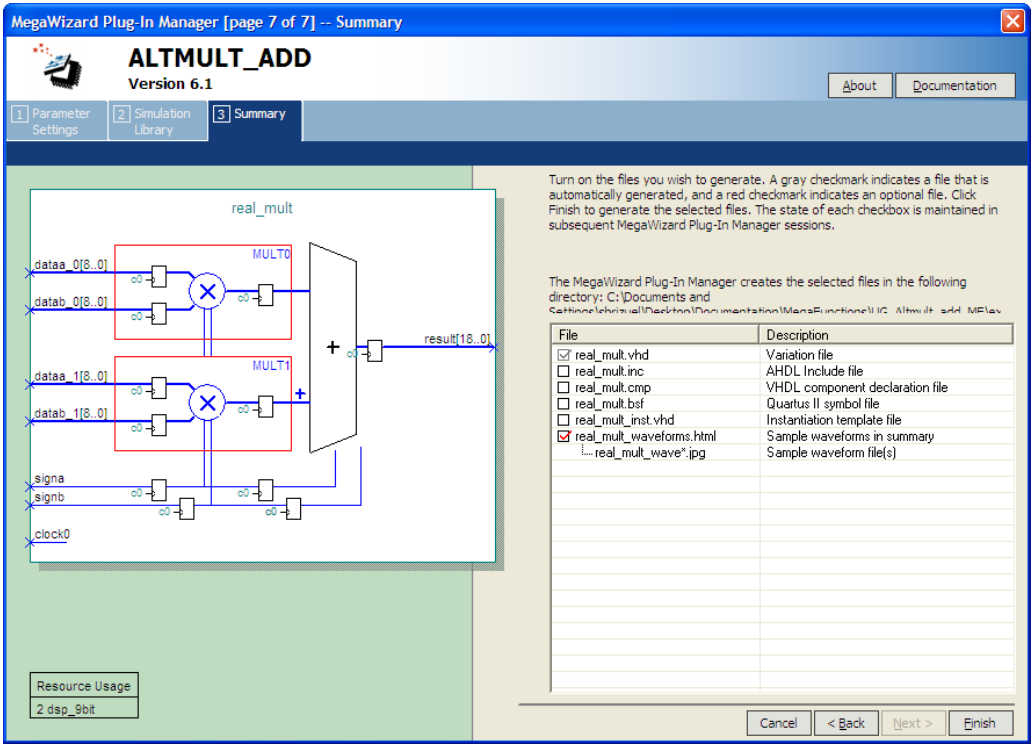
Table 2–3. altmult_add Plug-In Manager (Page 5) Options

Function	Description
Register input A of the multiplier	Select to create register for input A of the multiplier. Select 'More Options' to configure the register.
Register input B of the multiplier	Select to create register for input B of the multiplier. Select 'More Options' to configure the register.
What is the input A of the multiplier connected to?	Specify if input A of the multiplier is connected to a multiplier input or shiftin input.
What is the input B of the multiplier connected to?	Specify if input B of the multiplier is connected to a multiplier input or shiftin input.
Register output of the multiplier	Select to create a register for the output of the multiplier. Select 'More Options' to configure the register.

On page 7 of the altmult_add megafunction wizard, specify the types of files to be generated. You can choose from the HDL wrapper file, (<*function name*>**.v** | **.vhd** | **.tdf**), Block Symbol file (**.bsf**), instantiation template file (<*function name*>**_inst.v**), or Verilog Black Box declaration file (<*function name*>**_bb.v**) (Figure 2–6).

The HDL generated wrapper files (Verilog HDL, VHDL or AHDL) are selected automatically based on the choices you make on page 6 of the wizard.

Figure 2–6. MegaWizard Plug-In Manager —altmult_add [Page 7 of 7]



For more information about the ports for the altmult_add megafunction, refer to the *Specifications* chapter in this user guide

Inferring Megafunctions from HDL Code

Synthesis tools, including the Quartus II integrated synthesis, recognize certain types of HDL code and automatically infer the appropriate megafunction when a megafunction will provide optimal results. That is, the Quartus II software uses the Altera® megafunction code when compiling your design, even though you did not specifically instantiate the megafunction. The Quartus II software infers megafunctions because they are optimized for Altera devices, so the area and/or performance may be better than generic HDL code. Additionally, you must use megafunctions to access certain Altera architecture-specific features such as memory, DSP blocks, and shift registers, which generally provide improved performance compared with basic logic elements.

For more information, refer to the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*.

Instantiating Megafunctions in HDL Code

When you use the MegaWizard Plug-In Manager to set up and parameterize a megafunction, it creates either a VHDL or Verilog HDL wrapper file that instantiates the megafunction (a black-box methodology). For some megafunctions, you can generate a fully synthesizable netlist for improved results with EDA synthesis tools such as Synplify and Precision RTL Synthesis (a clear-box methodology). Both clear-box and black-box methodologies are described in the following third-party synthesis support chapters in the *Quartus II Handbook*:

- *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*
- *Quartus II Integrated Synthesis* chapter in volume 1 of the *Quartus II Handbook*
- *Synplicity Synplify and Synplify Pro Support* chapter in volume 1 of the *Quartus II Handbook*
- *Mentor Graphics Precision RTL Synthesis Support* chapter in volume 1 of the *Quartus II Handbook*

Identifying a Megafunction after Compilation

During compilation with the Quartus II software, analysis and elaboration is performed to build the structure of your design. Locate your megafunction in the Project Navigator window by expanding the compilation hierarchy and locating the megafunction by its name.

Similarly, to search for node names within the megafunction using the **Node Finder**, click the browse button in the **Look In** box and select the megafunction in the **Select Hierarchy Level** dialog box.

Simulation

The Quartus II Simulator provides an easy-to-use, integrated solution for performing simulations. The following sections describe the simulation options.

Quartus II Simulator

With the Quartus II Simulator, you can perform functional and timing simulations. A functional simulation in the Quartus II software enables you to verify the logical operation of your design without taking into consideration the timing delays in the FPGA. This simulation is performed using only your RTL code. When performing a functional simulation, you are able to view signals that exist before synthesis. You can find these signals with the **Registers: pre-synthesis**, **Design Entry**, or **Pin** filters in the **Node Finder**. The top-level ports of megafunctions are found using these three filters.

By contrast, timing simulations in the Quartus II software verify the operation of your design with annotated timing information. This simulation is performed using the post place-and-route netlist. When performing a timing simulation, you can view signals that exist after place and route. These signals are found with the **Post-Compilation** filter in the **Node Finder**. During synthesis and place-and-route, the names of your RTL signals change. Therefore, it might be difficult to find signals from your megafunction instantiation in the **Post-Compilation** filter. However, if you want to preserve the names of your signals during the synthesis and place-and-route stages, you must use the synthesis attributes `keep` or `preserve`. These are Verilog and VHDL synthesis attributes that direct analysis and synthesis to keep a particular wire, register, or node intact. You can use these synthesis attributes to keep a combinational logic node so you can observe the node during simulation.

For more information, refer to the *Quartus II Integrated Synthesis* chapter in volume 1 of the *Quartus II Handbook*.

EDA Simulation

Depending on which third-party simulation tool you are using, refer to the appropriate chapter in the *Simulation* section in volume 3 of the *Quartus II Handbook*. These tool-specific chapters show you how to perform functional and gate-level timing simulations that include the megafunctions, including the necessary files and directories where the files are located.

SignalTap II Embedded Logic Analyzer

The SignalTap® II Embedded Logic Analyzer provides you with a method of debugging all of the Altera megafunctions within your design. With the SignalTap II Embedded Logic Analyzer, you can capture and analyze data samples for the top-level ports of the Altera megafunctions in your design while your system is running at full speed.

To monitor signals from your Altera megafunctions, you must first configure the SignalTap II Embedded Logic Analyzer in the Quartus II software, then include the analyzer as part of your Quartus II project. The Quartus II software will then seamlessly embed the analyzer along with your design in the selected device.

For more information about using the SignalTap II Embedded Logic Analyzer, refer to the *Design Debugging Using the SignalTap II Embedded Logic Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

Design Example: Complex Multiplication

This design example uses the altmult_add megafunction to implement a complex multiplier. Consider the complex multiplication of $(A_0 + iA_1) \times (B_0 + iB_1)$. This expression can be expanded to $(A_0 \times B_0 \ A_1 \times B_1) + i(A_0 \times B_1 + A_1 \times B_0)$. To implement this, two multiplier/adder functions are required: one for the real expression and one for the imaginary expression.

Design Files

The design files are available in the literature page of the Altera web site under “user guides”.

Select the “Examples for altmult_add Megafunction User Guide” link from the examples page to download the design files.

Example 1

In this example, you perform the following activities:

- Create a complex multiplier using the altmult_add megafunction and the MegaWizard Plug-in Manager
- Implement the design and assign the EP1S10F780C5 device to the project
- Compile and simulate the design

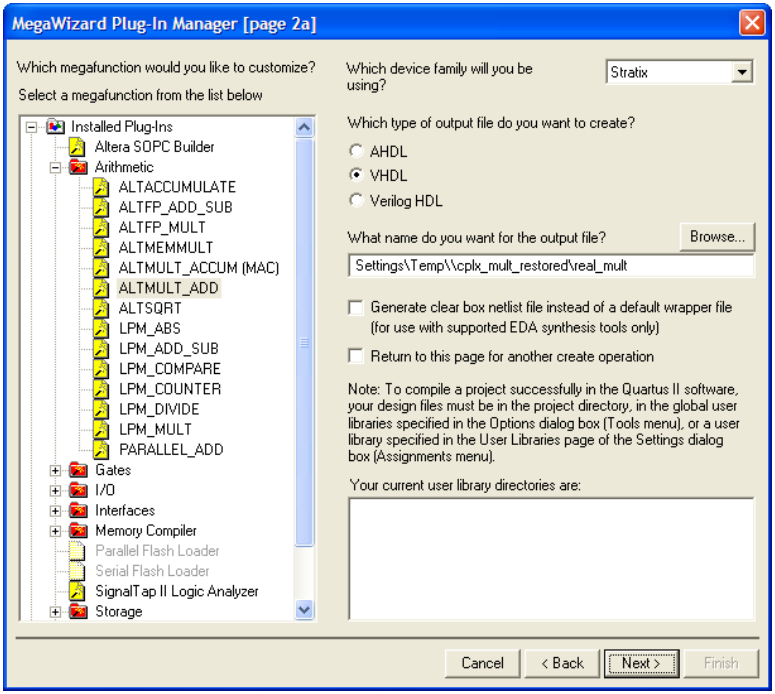
Generate a 9 × 9-Bit Multiplier/Adder (the Real Expression)

To generate a 9 × 9-bit multiplier/adder, perform the following steps:

1. Open the **altmult_add_DesignExample_ex1.zip** file and extract **cplx_mult.qar**. In the Quartus II software, open the **cplx_mult.qar** project and restore the archive file into your working directory.
2. On the Tools menu, click **MegaWizard Plug-In Manager**. The **MegaWizard Plug-In Manager** dialog box appears (Figure 2–1 on page 2–1).
3. Select **Create a new custom megafunction variation**, and click **Next**.
4. In the **Which megafunction would you like to customize?** list, click the + icon to expand **Arithmetic** and select **ALTMULT_ADD**.
5. In the **Which device family will you be using?** list, select **Stratix**.
6. Under **Which type of output file do you want to create?**, select **VHDL**.

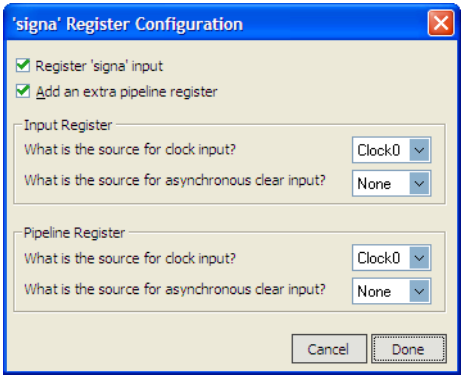
Specify the name of your output file as `real_mult`. Figure 2–7 shows the wizard after you have made these selections.

Figure 2–7. MegaWizard Plug-In Manager—altmult_add [Page 2a]


MegaWizard Plug-In Manager [page 2a]
Which megafunction would you like to customize?
Select a megafunction from the list below
Installed Plug-Ins
Altera SOPC Builder
Arithmetic
ALTACCUMULATE
ALTFP_ADD_SUB
ALTFP_MULT
ALTMEMMULT
ALTMULT_ACCUM (MAC)
ALTMULT_ADD
ALTSQRT
LPM_ABS
LPM_ADD_SUB
LPM_COMPARE
LPM_COUNTER
LPM_DIVIDE
LPM_MULT
PARALLEL_ADD
Gates
I/O
Interfaces
Memory Compiler
Parallel Flash Loader
Serial Flash Loader
SignalTap II Logic Analyzer
Storage
Which device family will you be using?
Stratix
Which type of output file do you want to create?
AHDL
VHDL
Verilog HDL
What name do you want for the output file?
Browse...
Settings\Temp\\cplx_mult_restored\real_mult
Generate clear box netlist file instead of a default wrapper file (for use with supported EDA synthesis tools only)
Return to this page for another create operation
Note: To compile a project successfully in the Quartus II software, your design files must be in the project directory, in the global user libraries specified in the Options dialog box (Tools menu), or a user library specified in the User Libraries page of the Settings dialog box (Assignments menu).
Your current user library directories are:
Cancel
< Back
Next >
Finish

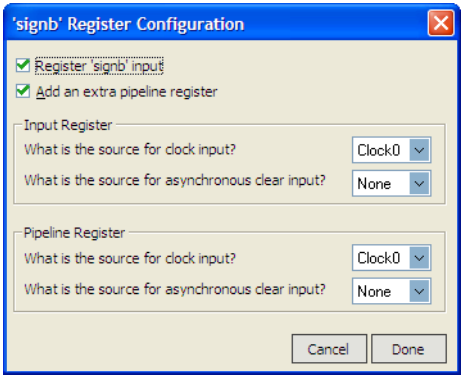
7. Click **Next**.
8. In the **What is the number of multipliers?** list, select **2**.
9. Turn on **All multipliers have similar configurations**.
10. In both the **How wide should the A input buses be?** and **How wide should the B input buses be?** lists, select **9**.
11. In the **How wide should the 'result' output bus be?** list, select **19**.
12. Turn off **Create a 4th asynchronous clear input option** and **Create an associated clock enable for each clock**.
13. In the **What is the representation format for A inputs?** option, select **Variable**.
14. Click **More Options** for 'signa'. The **'signa' Register Configuration** dialog box appears (Figure 2–8).

Figure 2–8. 'signa' Register Configuration



15. Turn on **Register 'signa' input** and **Add an extra pipeline register**.
16. Under both **input Register** and **Pipeline Register**, select the following options:
    - In the **What is the source for clock input**? option, select **Clock0**.
    - In the **What is the source for asynchronous clear input?** option, select **None**.
17. Click **Done**.
18. In the **What is the representation format for B inputs** option, select **Variable**.
19. Click **More Options** for 'signb'. The **'signb' Register Configuration** dialog box appears (Figure 2–9).

Figure 2–9. 'signb' Register Configuration



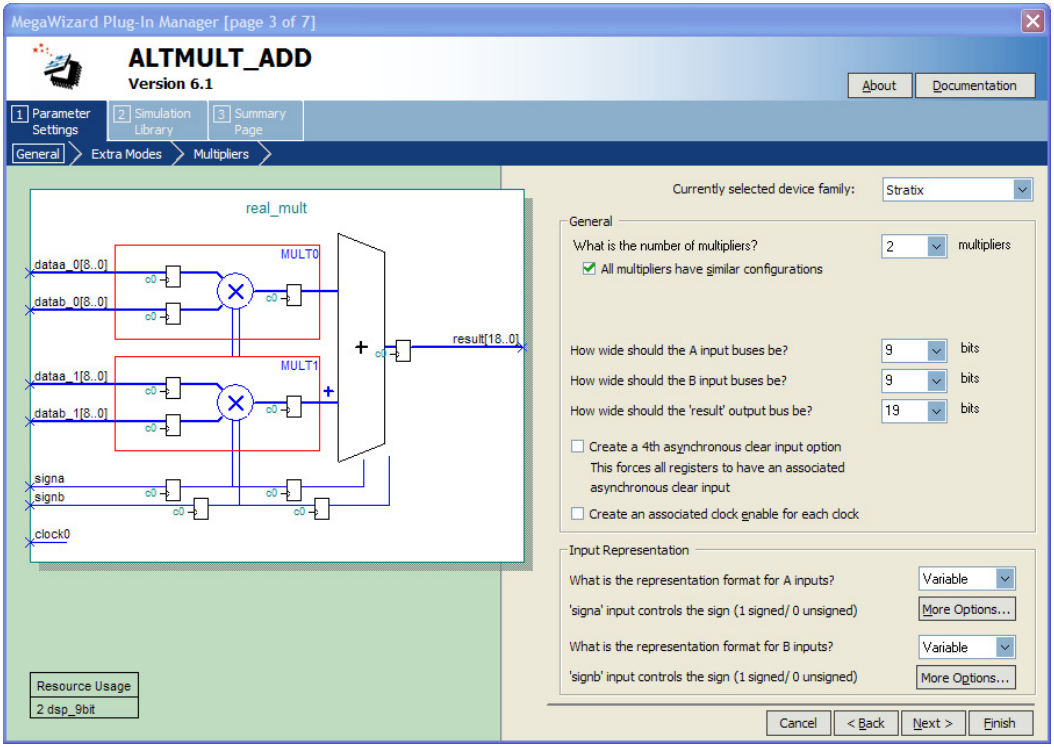
20. Turn on **Register 'signb' input** and **Add an extra pipeline register**.

21. Under both **Input Register** and **Pipeline Register**, select the following options:

    - In the **What is the source for clock input**? option, select **Clock0**.
    - In the **What is the source for asynchronous clear input?** option, select **None**.

22. Click **Done**. Figure 2–10 shows the wizard after you have made these selections.

Figure 2–10. MegaWizard Plug-In Manager—altmult_add [Page 3 of 7]

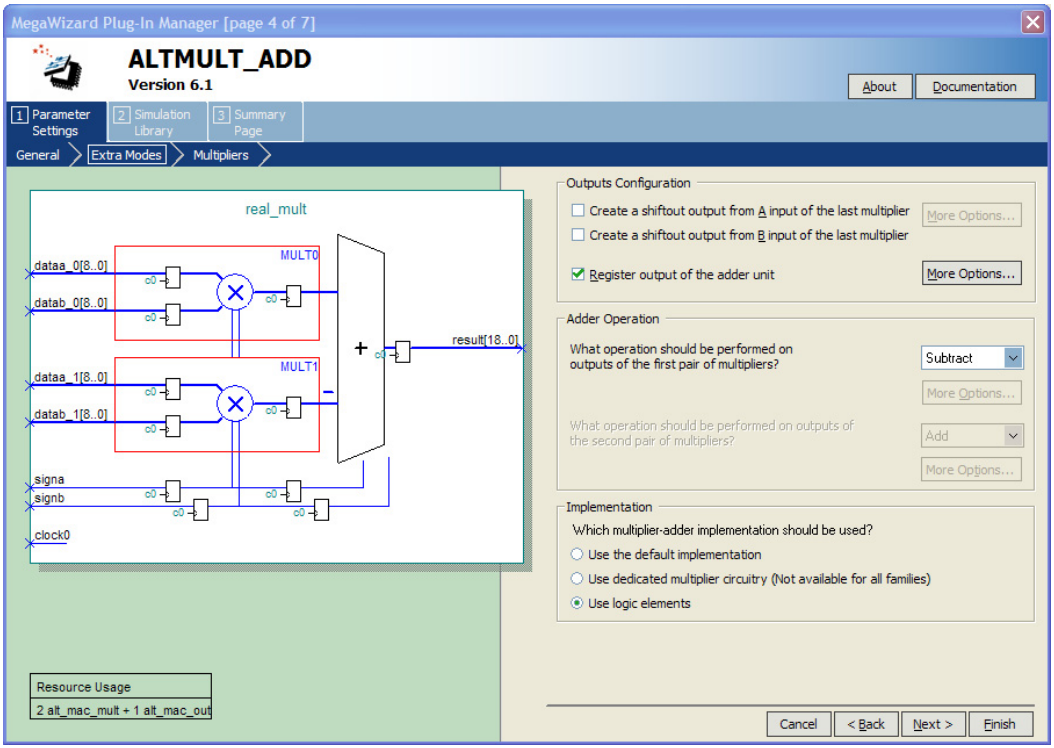


23. Click **Next**.
24. Under **Outputs Configuration**, turn off **Create a shiftout output from A input of the last multiplier** and **Create a shiftout output from B input of the last multiplier**.
25. Turn on **Register output of the adder unit**.
26. Click **More Options** for **Register output of the adder unit**. The **Output Register Configuration** dialog box appears.
27. In the **What is the source for clock input**? option, select **Clock0**.
28. In the **What is the source for asynchronous clear input?** option, select **None**.
29. Click **Done**.

30. Under **Adder Operation**, in the **What operation should be performed on outputs of the first pair of multipliers?** list, select **Subtract**.

31. Under **implementation**, select **Use logic elements**. Figure 2–11 shows the wizard after you have made these selections.

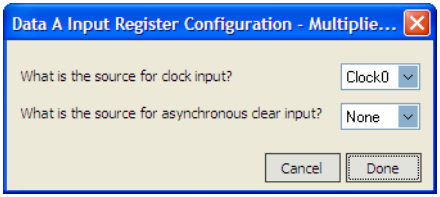
Figure 2–11. MegaWizard Plug-In Manager—altmult_add [Page 4 of 7]



32. Click **Next**.

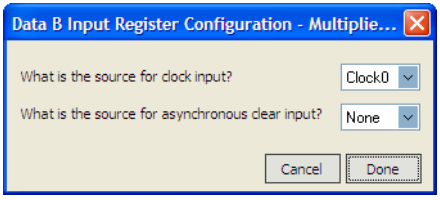
33. Under **Input Configuration**, turn on **Register input A of the multiplier** and click **More Options**. The **Data A Input Register Configuration - Multiplier 0** dialog box appears (Figure 2–12).

Figure 2–12. Data A Input Register Configuration - Multiplier 0



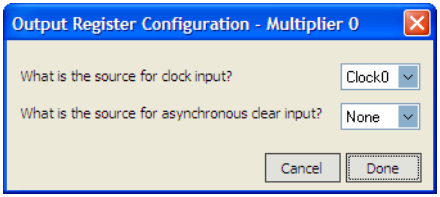
34. In the **What is the source for clock input?** list, select **Clock0**.
35. In the **What is the source for asynchronous clear input?** list, select **None**.
36. Click **Done**.
37. Under **Input Configuration**, turn on the **Register input B of the multiplier** option. Click **More Options**. The **Data B input Register Configuration - Multiplier 0** dialog box appears (Figure 2–13).

Figure 2–13. Data B Input Register Configuration - Multiplier 0



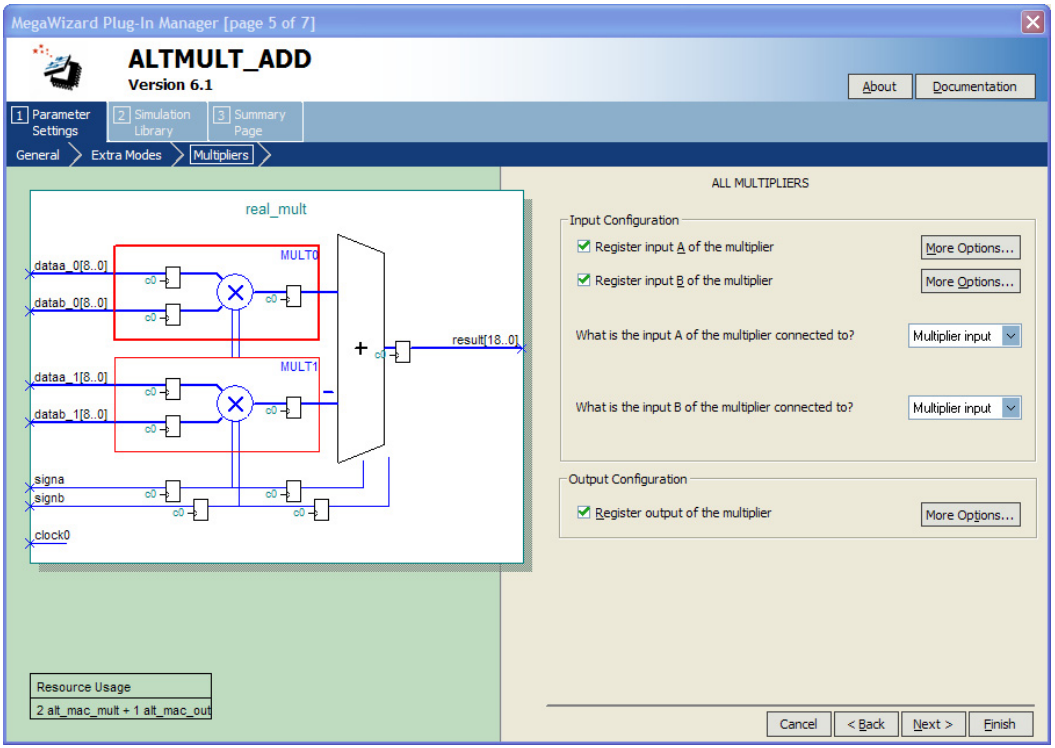
38. In the **What is the source for clock input?** list, select **Clock0**.
39. In the **What is the source for asynchronous clear input?** list, select **None**.
40. Click **Done**.
41. Under **Input Configuration**, in both the **What is the input A of the multiplier connected to?** and **What is the input B of the multiplier connected to?** lists, select **Multiplier input**.
42. Under **Output Configuration**, turn on **Register output of the multiplier**. Click **More Options**. The **Output Register Configuration - Multiplier 0** dialog box appears (Figure 2–14).

Figure 2–14. Output Register Configuration - Multiplier 0



43. In the **What is the source for clock input?** list, select **Clock0**.
44. In the **What is the source for asynchronous clear input?** list, select **None**.
45. Click **Done**. Figure 2–15 shows the wizard after you have made these selections.

Figure 2–15. MegaWizard Plug-In Manager—altmult_add [Page 5 of 7]



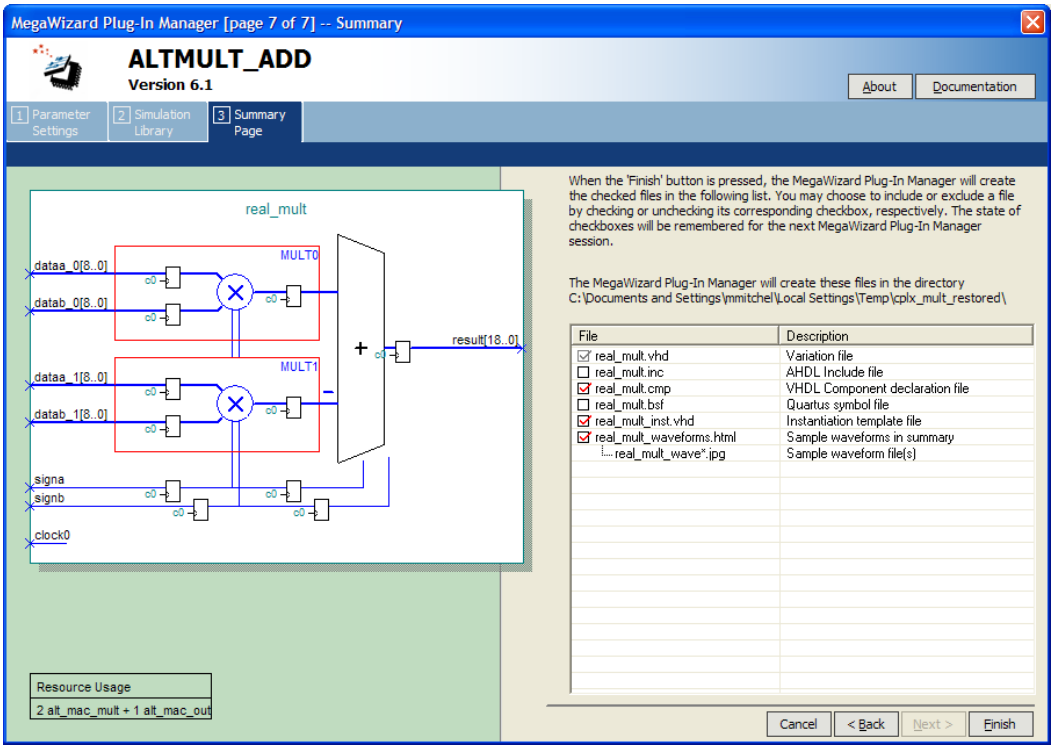
46. Click **Finish**.

    The final page of the wizard shows the file that is generated for your custom megafunction variation. The gray check marks indicate files that are always generated; the other files are optional and are generated only if selected (indicated by a red check mark). Turn on the boxes to select the files that you want generated.

47. Turn on **VHDL Component declaration file** and **Instantiation template file**.

48. Turn off **Quartus symbol file** and **AHDL include file**. Leave the other options in their default settings. Figure 2–16 shows the wizard after you have made these selections.

Figure 2–16. MegaWizard Plug-In Manager—altmult_add [Page 7 of 7], Summary



49. Click **Finish**.

The altmult_add variation is now built.

Generate a 9 x 9-Bit Multiplier/Adder (the Imaginary Expression)

1. In the Tools menu, click **MegaWizard Plug-In Manager**. The **MegaWizard Plug-In Manager [page 1]** dialog box appears (Figure 2–1 on page 2–1).

2. Select **Create a new custom megafunction variation** and click **Next**.

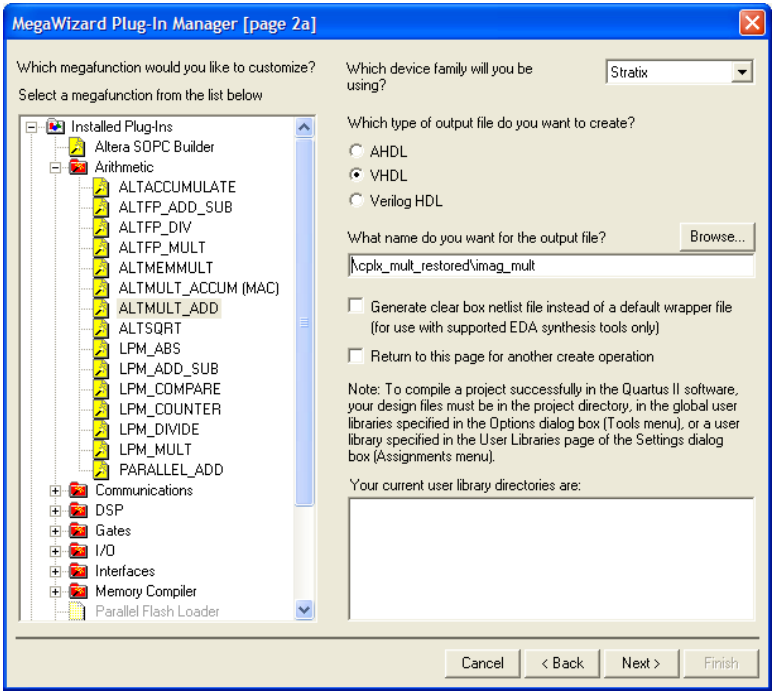
3. In the **Which megafunction would you like to customize?** list, click the + icon to expand **Arithmetic** and select **ALTMULT_ADD**.

4. In the **Which device family will you be using?** list, select **Stratix**.

5. Under **Which type of output file do you want to create?**, select **VHDL**.

6. Specify the output file name as **imag_mult**. Figure 2–17 shows the wizard after you have made these selections.

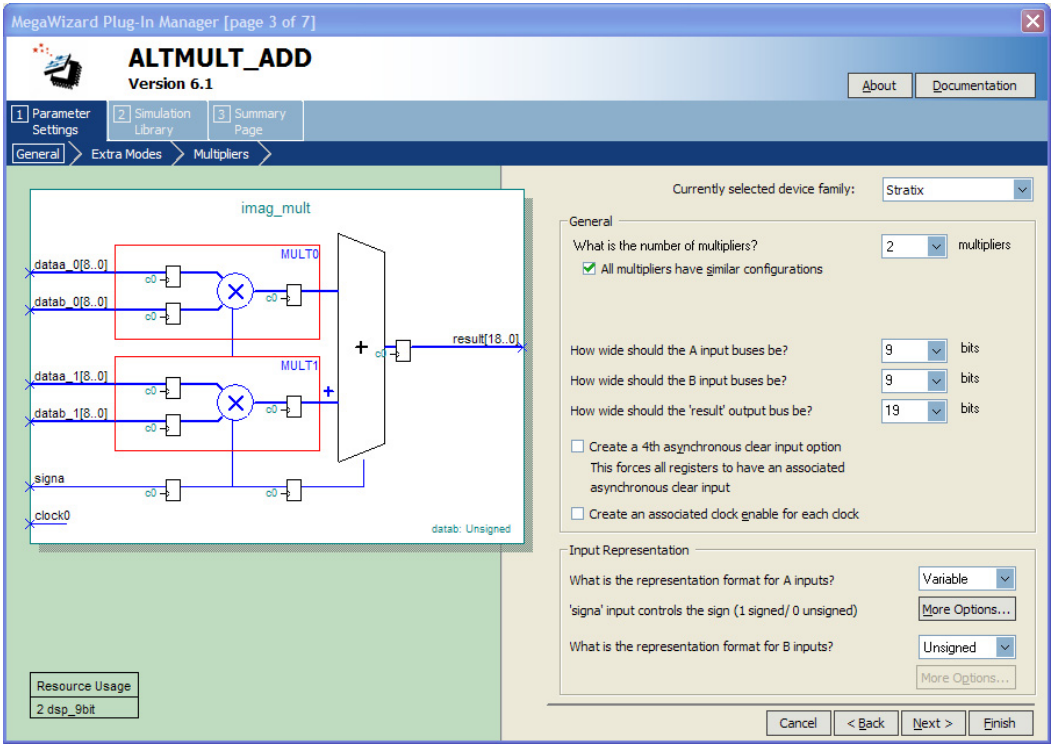
Figure 2–17. MegaWizard Plug-In Manager—altmult_add [Page 2a]



7. Click **Next**.

8. In the **What is the number of multipliers?** list, select **2**.
9. Turn on **All multipliers have similar configurations** option.
10. In both the **How wide should the A input buses be?** and **How wide should the B input buses be?** lists, select **9**.
11. In the **How wide should the 'result' output bus be?** list, select **19**.
12. Turn off the **Create a 4th asynchronous clear input option** and **Create an associated clock enable for each clock**.
13. In the **What is the representation format for A inputs** list, select **Variable**.
14. Click **More Options** for 'signa'. The **'signa' Register Configuration** dialog box appears.
15. Turn on **Register 'signa' input** and **Add an extra pipeline register**.
16. Under i**nput Register** and **Pipeline Register**, select the following options:
    - In the **What is the source for clock input?** list, select **Clock0**.
    - In the **What is the source for asynchronous clear input?** list, select **None**.
17. Click **Done**.
18. In the **What is the representation format for B inputs** list, select **Variable**.
19. Click **More Options** for 'signb'. The **'signb' Register Configuration** dialog box appears.
20. Turn on **Register 'signb' input** and **Add an extra pipeline register**.
21. Under i**nput Register** and **Pipeline Register**, select the following options:
    - In the **What is the source for clock input**? list, select **Clock0**.
    - In the **What is the source for asynchronous clear input?** list, select **None**.
22. Click **Done**. Figure 2–18 shows the wizard after you have made these selections.

Figure 2–18. MegaWizard Plug-In Manager [Page 3 of 7]



23. Click **Next**.
24. Under **Outputs Configuration**, turn off **Create a shiftout output from A input of the last multplier** and **Create a shiftout output from B input of the last multiplier**.
25. Turn on **Register output of the adder unit**. Click **More Options**. The **Output Register Configuration** dialog box appears.
26. In the **What is the source for clock input**? list, select **Clock0**.
27. In the **What is the source for asynchronous clear input?** list, select **None**.
28. Click **Done**.
29. Under **Outputs configuration**, turn on **Register output of the adder unit**.

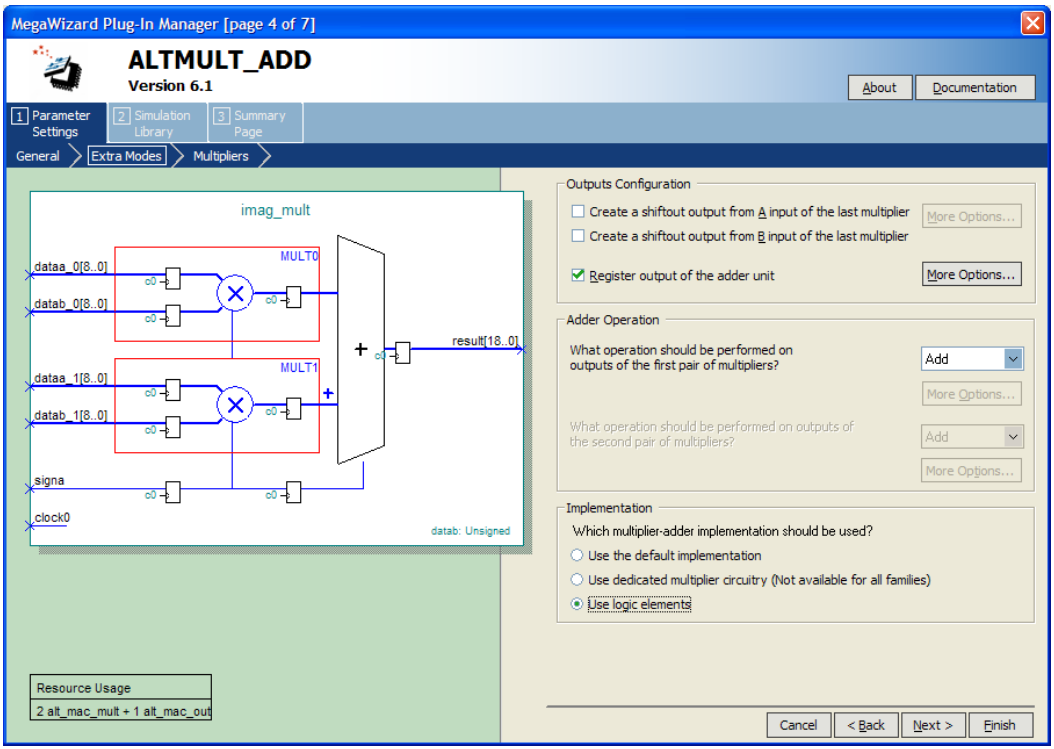
30. Under **Outputs configuration**, turn on **Register output of the adder unit**.

31. Under **Adder Operation**, in the **What operation should be performed on outputs of the first pair of multipliers?** list, select **Add**.

32. Under **Implementation**, select **Use logic elements**.

    Figure 2–19 shows the wizard after you have made these selections.

Figure 2–19. MegaWizard Plug-In Manager [Page 4 of 7]



33. Click **Next**.

34. Under **Input Configuration**, turn on **Register input A of the multiplier** and click **More Options**. The **Data A input Register Configuration - Multiplier 0** dialog box appears.

35. In the **What is the source for clock input?** list, select **Clock0**.

36. In the **What is the source for asynchronous clear input?** list, select **None**.

37. Click **Done**.

38. Under **Input Configuration**, turn on **Register input B of the multiplier** and click **More Options**. The **Data B input Register Configuration - Multiplier 0** dialog box appears.

39. In the **What is the source for clock input?** list, select **Clock0**.

40. In the **What is the source for asynchronous clear input?** list, select **None**.

41. Click **Done**.

42. Under **Input Configuration**, select **Multiplier input** from both the **What is the input A of the multiplier connected to?** and **What is the input B of the multiplier connected to?** lists.

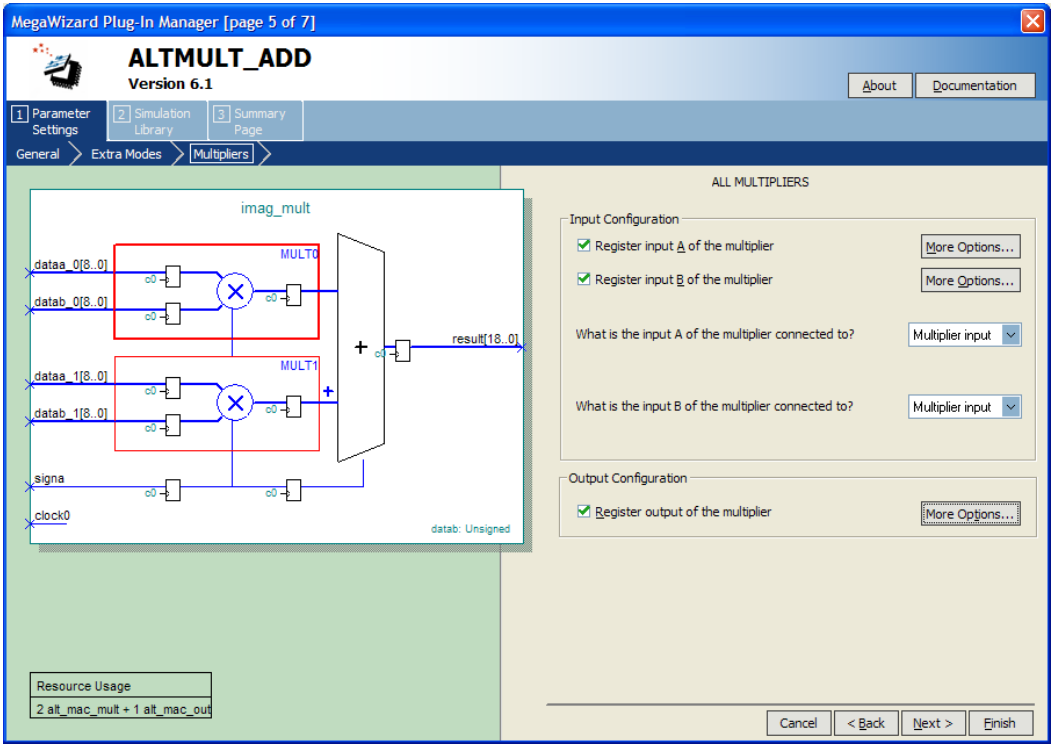
43. Under **Output Configuration**, turn on **Register output of the multiplier** and click **More Options**. The **Output Register Configuration** dialog box appears.

44. In the **What is the source for clock input?** list, select **Clock0**.

45. In the **What is the source for asynchronous clear input?** list, select **None**.

46. Click **Done**. Figure 2–20 shows the wizard after you have made these selections.

Figure 2–20. MegaWizard Plug-In Manager [Page 5 of 7]



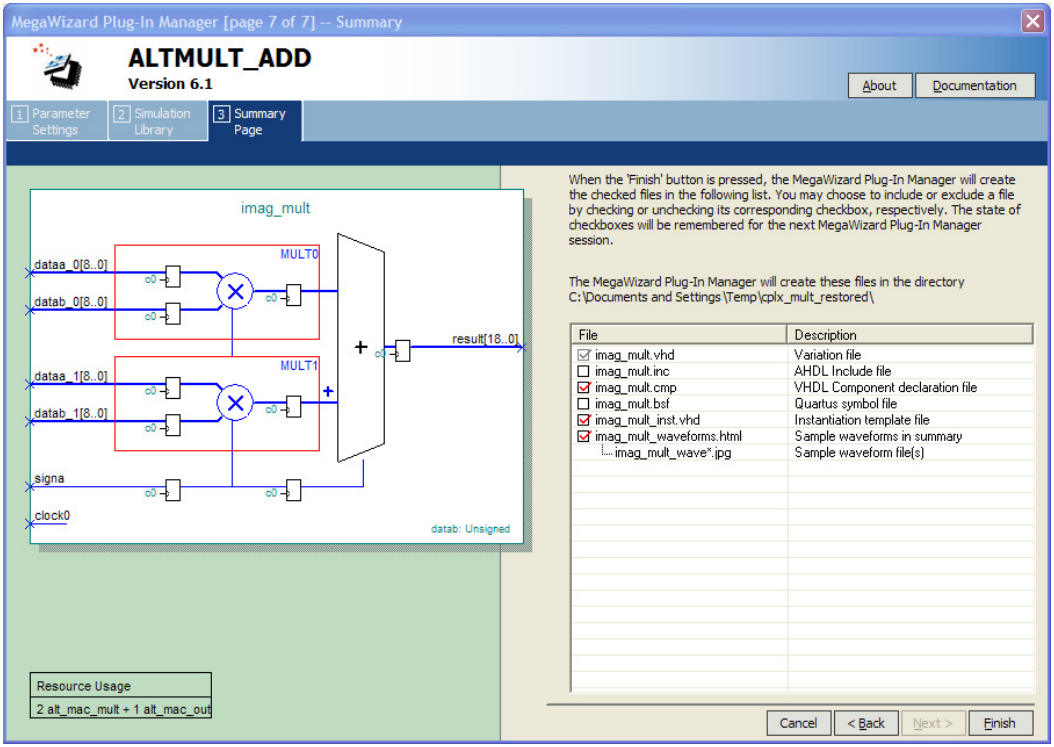
47. Click **Finish**.

    The final page of the wizard shows the files that are generated for your custom megafunction variation. The gray check marks indicate files that are always generated; the other files are optional and are generated only if selected (indicated by a red check mark). Turn on the boxes to select the files that you want generated.

48. Turn on **VHDL Component declaration file** and **Instantiation template file**.

49. Turn off **Quartus symbol file** and **AHDL Include file**. Leave the other options in their default settings. Figure 2–21 shows the wizard after you have made these selections.

Figure 2–21. MegaWizard Plug-In Manager—altmult_add [Page 7 of 7], Summary



50. Click **Finish**.

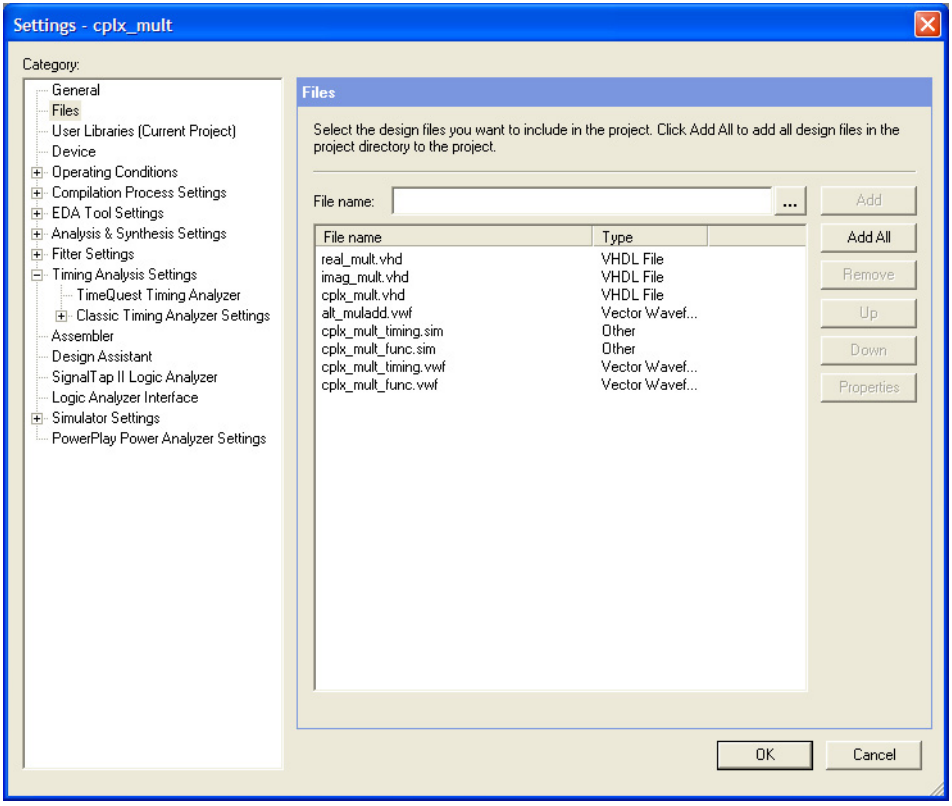
The altmult_add variation is now built.

Combine real_mult and image_mult to Create a Complex Multiplier

This section describes how to create a new top-level Verilog HDL file in the Quartus II software.

1. From your working directory, open the **cplx_mult.vhd** file.
2. On the Project menu, click **Add/Remove Files in Project**. The **Settings** dialog box appears (Figure 2–22).

Figure 2–22. Settings Dialog Box

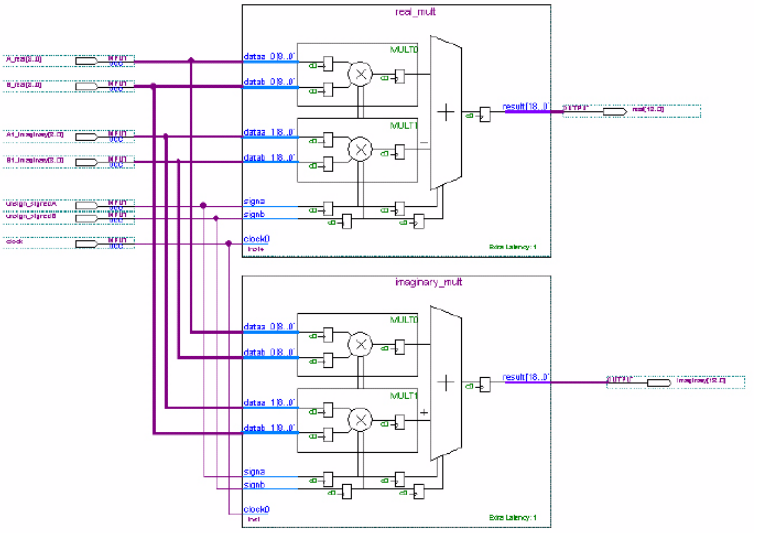


3. Click browse to select the **cplx_mult.vhd** file in the project folder, click **Open**, and click **Add** in the Settings dialog box to add the file to the project.

4. Click **OK**.

The top-level file is now added to the project. You have now created a complete design file (Figure 2–23). (Note that this image is extracted from the symbol file, which we have not created in this design example.)

Figure 2–23. Complex Multiplier Design File

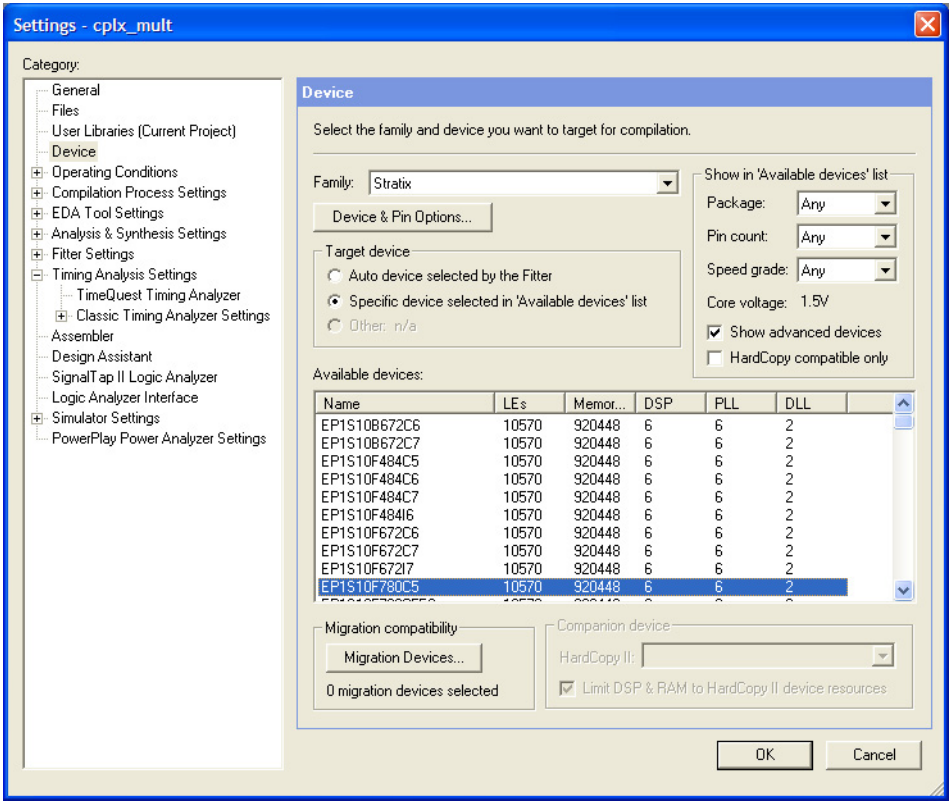


Implement the 9 × 9 Complex Multiplier Design

This section describes how to assign the EP1S10F780C5 device to the project and compile the project in the Quartus II software.

1. On the Assignment menu, click **Device**. The **Settings** dialog box appears (Figure 2–24).

Figure 2–24. Settings Dialog Box



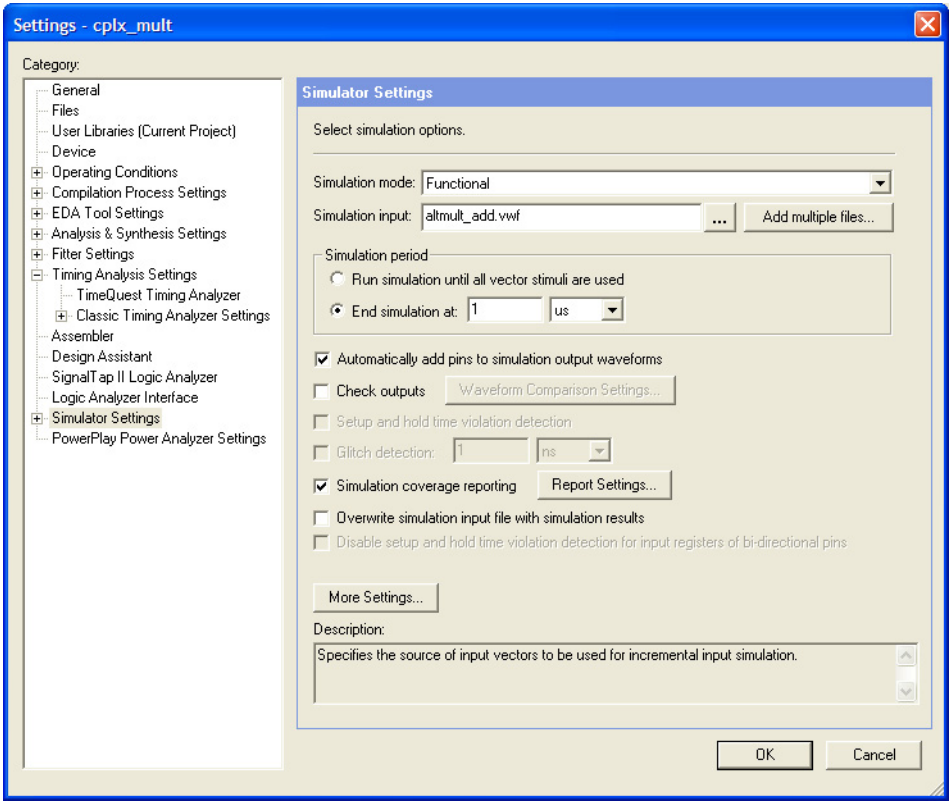
2. In the **Family** list, select **Stratix**.
3. Under **Target device**, select **Specific device selected in 'Available devices' list**.
4. In the **Available devices** list, select **EP1S10F780C5**.
5. Click **OK**.
6. In the Processing menu, click **Start Compilation** to compile the design.
7. When the **Full compilation was successful** box appears, click **OK**.

Functional Results—Simulate the Complex Multiplier Design

This section describes how to verify the design example you just created by simulating the design using the Quartus II Simulator. To set up the Quartus II Simulator, perform the following steps:

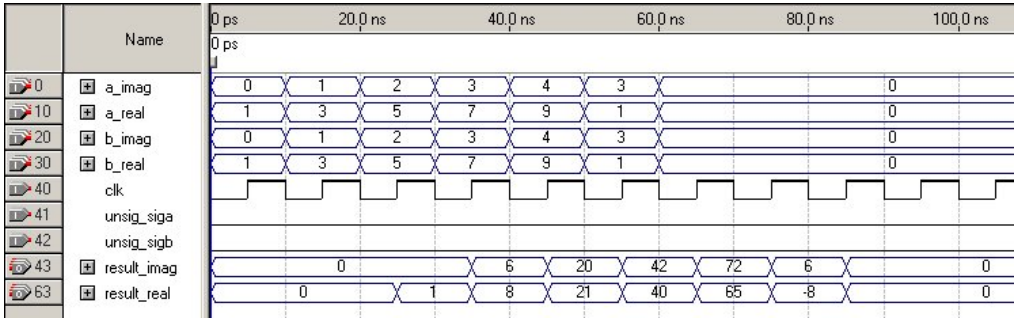
1. On the Processing menu, select **Generate Functional Simulation Netlist**.
2. When the **Functional Simulation Netlist Generation was successful** box appears, click **OK**.
3. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
4. In the **Category** list, select **Simulator Settings**. The **Simulator Settings** page appears.
5. In the **Simulation mode** list, select **Functional**.
6. In the **Simulation Input** box, type `altmult_add.vwf`, or click the browse to select the file in the project folder.
7. Under **Simulation period**, select **End simulation at**, type `1`, and select **us**.
8. Turn on **Automatically add pins to simulation output waveforms** and **Simulation coverage reporting**.
9. Turn off **Check outputs**.
10. Turn off **Overwrite simulation input file with simulation results**. Figure 2–25 shows the **Settings** dialog box after you have made these selections.

Figure 2–25. Functional Simulation Settings



11. In the Category list, click the + icon to expand **Simulator Settings**.
12. Select **Simulation Power**. The **Simulation Power** page appears.
13. Turn off **Generate Signal Activity File** and **Generate VCD File**.
14. Click **OK**.
15. On the Processing menu, Click **Start Simulation** to run simulation.
16. The **Simulation was successful** box appears. Click **OK**.
17. The Simulation Report window appears. Verify the simulation waveform results (Figure 2–26).

Figure 2–26. Simulation Waveform

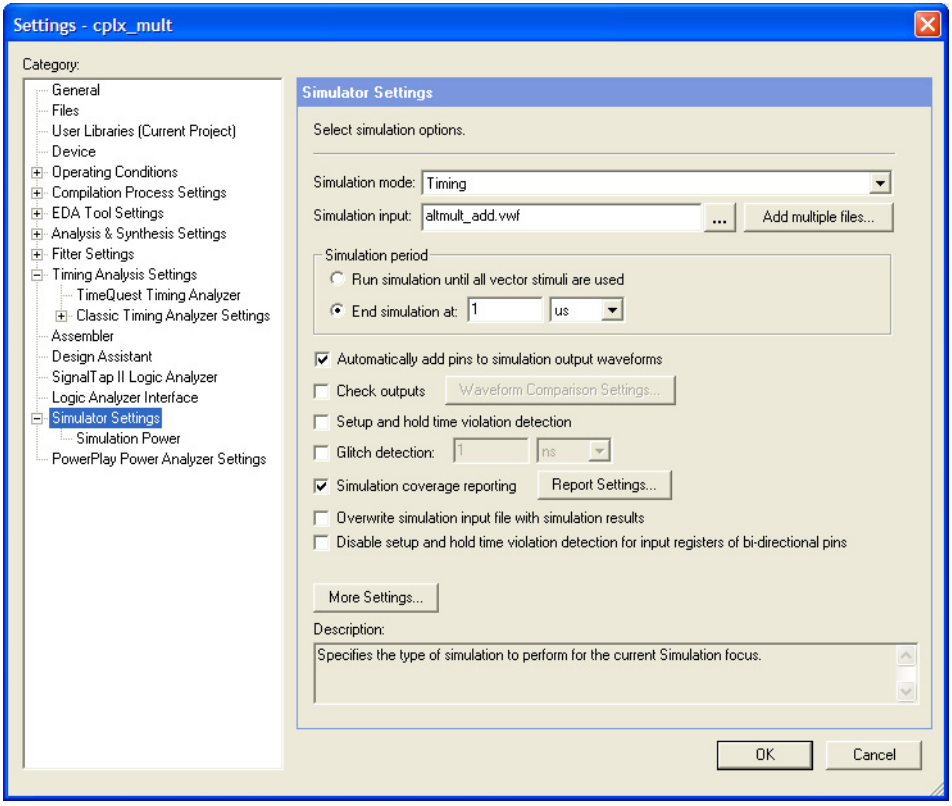


Timing Results

This section describes how to verify the timing after implementation using the Quartus II Simulator. To set up the Quartus II Simulator, perform the following steps in the Quartus II software:

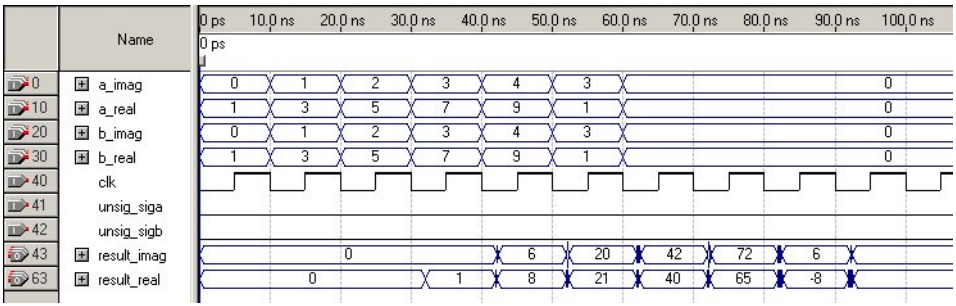
1. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
2. In the **Category** list, select **Simulator Settings**. The **Simulator Settings** page appears.
3. In the **Simulation mode** list, select **Timing**.
4. In the **Simulation Input** box, type `altmult_add.vwf`, or click **Browse** (**...**) to select the file in the project folder.
5. Under **Simulation period**, select **End simulation at**, type `1`, and select **us**.
6. Turn on **Automatically add pins to simulation output waveforms** and **Simulation coverage reporting**.
7. Turn off **Check outputs**, **Setup and hold time violation detection**, **Glitch detection**, and **Overwrite simulation input file with simulation results**. Figure 2–27 shows the **Settings** dialog box after you have made these selections.

Figure 2–27. Simulator Settings Page



8. In the **Category** list, click the + icon to expand **Simulator Settings**.
9. Select **Simulation Power**. The **Simulation Power** page appears.
10. Turn off **Generate Signal Activity File** and **Generate VCD File**.
11. Click **OK**.
12. On the Processing menu, Click **Start Simulation** to run simulation.
13. The **Simulation was successful** box appears. Click **OK**.
14. The Simulation Report window appears. Verify the simulation waveform results (Figure 2–28).

Figure 2–28. Timing Simulation Waveform



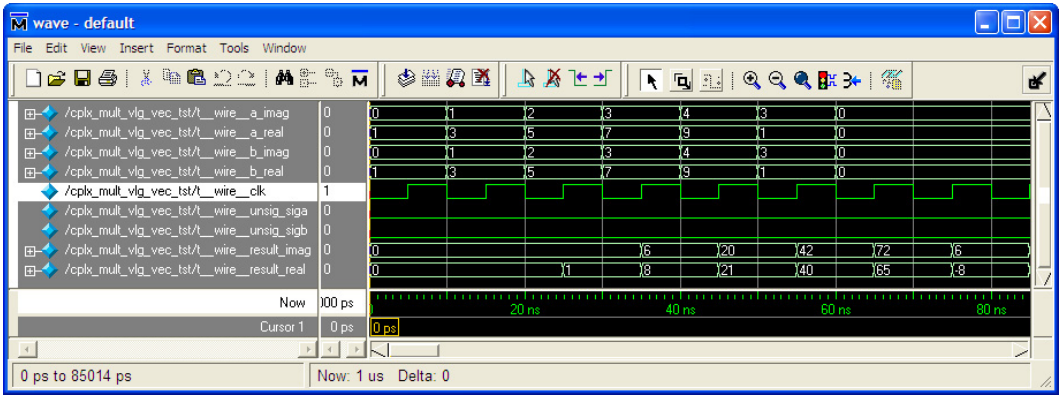
Simulate the Complex Multiplier Design in ModelSim-Altera

To simulate the design in ModelSim compare the results of both simulators. Setup the ModelSim-Altera simulator by performing the following steps:

1. Unzip **almult_add_ex1_msim.zip** to any working directory on your PC.
2. Start ModelSim-Altera.
3. On the File menu, click **Change Directory**. The **Choose folder** dialog box appears and select the folder where you have unzipped your files. Click **OK**.
4. On the Tools menu, click **Execute Macro**. The **Execute Do File** dialog box appears.
5. Select **cplx_mult_functional.do** and click **Open**.

cplx_mult_functional.do is a script file for ModelSim that automates all the necessary settings for the functional simulation. You can verify the results in the Waveform Viewer window (Figure 2–29).

Figure 2–29. ModelSim Waveform Viewer for Functional Simulation



If needed, you can rearrange signals, remove redundant signals, and change the radix to suit the results in the Quartus II Simulator.

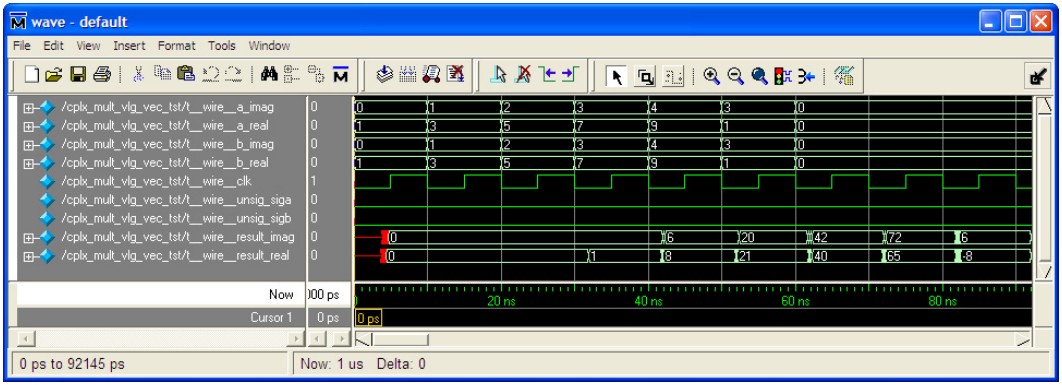
To further check the timing simulation, perform the following steps:

1. In the Tools menu, click **Execute Macro**. The **Execute Do File** dialog box appears.

2. Select **cplx_mult_timing.do** and click **OK**.

cplx_mult_timing.do is a script file for ModelSim that automates all the necessary settings for the timing simulation. You can verify the results in the Waveform Viewer window (Figure 2–30).

Figure 2–30. ModelSim Waveform Viewer for Timing Simulation



If needed, you can rearrange signals, remove redundant signals, and change the radix to suit the results in the Quartus II Simulator.

Design Example: Implementing a Simple FIR Filter

This design example uses the altmult_add megafunction to implement a FIR filter in the form shown in Equation .

$$y(t) = \sum_{i=0}^{n-1} A(t-i)B(i)$$

In this equation, *n* represents the number of taps, *A*(*t*) represents the sequence of input samples, and *B*(*t*) represents the filter coefficients.

This example implements a simple FIR filter with *n* = 4, which is called a 4-tap filter. The number of taps (*n*) can be any value, but the example shown in Equation shows the FIR filter with four taps. To implement this filter, the coefficients of data `B` is loaded into the `B` registers in parallel and a `shiftin` register moves data `A(0)` to `A(1)` to `A(2)`, and so on. With a 4-tap filter, at a given time [T], the sum of four products is computed. This function is implemented using the shift register chain option in the altmult_add megafunction. In the example shown in Equation , input *B* represents the coefficients and data *A* represents the data that is shifted into. The *A* input (data) is shifted in with the main clock, `clock0`. The *B* input (coefficients) is loaded with `clock1` rising edge and enable signal held high.

Design Files

The design files are available in the Quartus II Projects section on the Design Examples page of the Altera web site:

http://www.altera.com/support/examples/quartus/quartus.html

Select the "Examples for altmult_add Megafunction User Guide" link from the examples page to download the design files.

Example 2

This design example uses the altmult_add megafunction to create a multiplier-add that targets the default implementation which goes into DSP blocks. In this example, you perform the following activities:

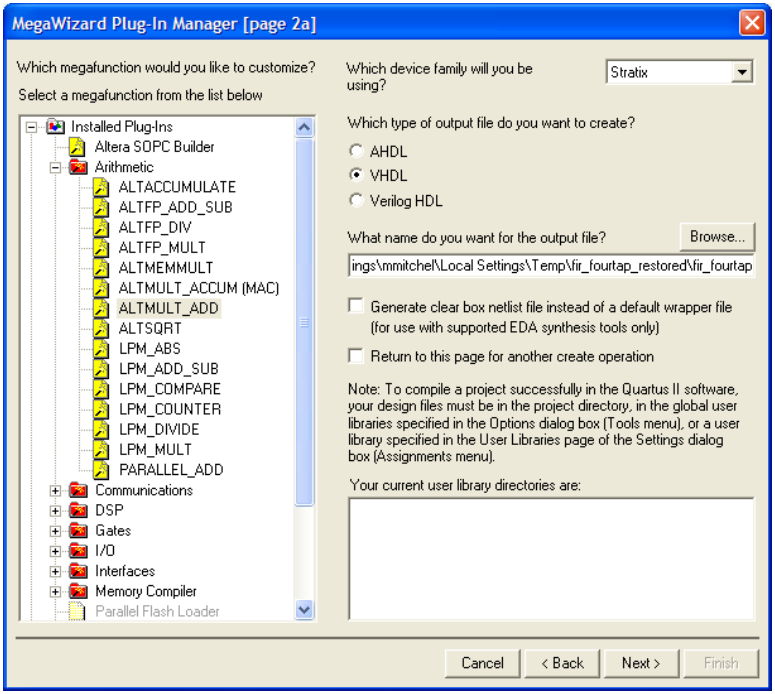
- Create a FIR filter using the altmult_add megafunction and the MegaWizard Plug-in Manager
- Implement the design and assign the EP1S10F780C5 device to the project
- Compile and simulate the design

Generate a 4-Tap FIR Filter

To generate a 4-tap FIR filter, perform the following steps:

1. Open the **altmult_add_DesignExample_ex2.zip** file and extract **fir_fourtap.qar**. In the Quartus II software, open the **fir_fourtap.qar** project and restore the archive file into your working directory.
2. In the Tools menu, click **MegaWizard Plug-In Manager**. The **MegaWizard Plug-In Manager** dialog box appears (Figure 2–1 on page 2–1).
3. Select **Create a new custom megafunction variation** and click **Next**.
4. In the **Which megafunction would you like to customize?** list, click the + icon to expand **Arithmetic** and select **ALTMULT_ADD**.
5. In the **Which device family will you be using?** list, select **Stratix**.
6. Under **Which type of output file do you want to create?**, select **VHDL**.
7. In the **What name do you want for the output file?**, enter `fir_fourtap`, or click **Browse** (**...**) to select the filename. Figure 2–31 shows the wizard after you have made these selections.

Figure 2–31. MegaWizard Plug-In Manager


MegaWizard Plug-In Manager [page 2a]
Which megafunction would you like to customize?
Select a megafunction from the list below
Installed Plug-Ins
Altera SOPC Builder
Arithmetic
ALTACCUMULATE
ALTFP_ADD_SUB
ALTFP_DIV
ALTFP_MULT
ALTMEMMULT
ALTMULT_ACCUM (MAC)
ALTMULT_ADD
ALTSQRT
LPM_ABS
LPM_ADD_SUB
LPM_COMPARE
LPM_COUNTER
LPM_DIVIDE
LPM_MULT
PARALLEL_ADD
Communications
DSP
Gates
I/O
Interfaces
Memory Compiler
Parallel Flash Loader
Which device family will you be using?
Stratix
Which type of output file do you want to create?
AHDL
VHDL
Verilog HDL
What name do you want for the output file?
Browse...
ings\mmitchel\Local Settings\Temp\fir_fourtap_restored\fir_fourtap
Generate clear box netlist file instead of a default wrapper file (for use with supported EDA synthesis tools only)
Return to this page for another create operation
Note: To compile a project successfully in the Quartus II software, your design files must be in the project directory, in the global user libraries specified in the Options dialog box (Tools menu), or a user library specified in the User Libraries page of the Settings dialog box (Assignments menu).
Your current user library directories are:
Cancel
< Back
Next >
Finish

8. Click **Next**.
9. In the **What is the number of multipliers?** list, select **4**.
10. Turn on **All multipliers have similar configurations**.
11. In both the **How wide should the A input busses be?** and **How wide should the B input busses be?** lists, select **16**.
12. In the **How wide should the 'result' output bus be?** list, select **34**.
13. Turn off the **Create a 4th asynchronous clear input option**.
14. Turn on **Create an associated clock enable for each clock**.
15. In the **What is the representation format for A inputs** list, select **Variable**.

16. Click **More Options** for 'signa'. The **'signa' Register Configuration** dialog box appears.

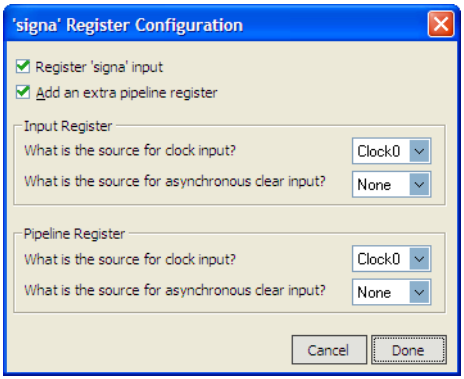
17. Turn on **Register 'signa' input** and **Add an extra pipeline register**.

18. Under both **input Register** and **Pipeline Register**, select the following options:

    - In the **What is the source for clock input**? list, select **Clock0**.
    - In the **What is the source for asynchronous clear input?** list, select **None**.

    Figure 2–32 shows the **'signa' Register Configuration** dialog box after you have made these selections.

Figure 2–32. 'signa' Register Configuration Dialog Box



19. Click **Done**.

20. In the **What is the representation format for B inputs** list, select **Variable**.

21. Click **More Options** for 'signb'. The **'signb' Register Configuration** dialog box appears.

22. Turn on the **Register 'signb' input** and **Add an extra pipeline register** options.

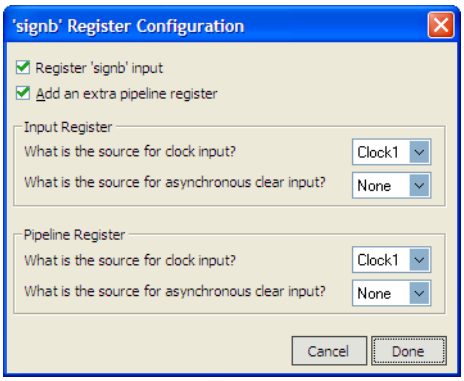
23. Under both **input Register** and **Pipeline Register**, select the following options:

    - In the **What is the source for clock input**? list, select **Clock1**.

- In the **What is the source for asynchronous clear input?** list, select **None**.

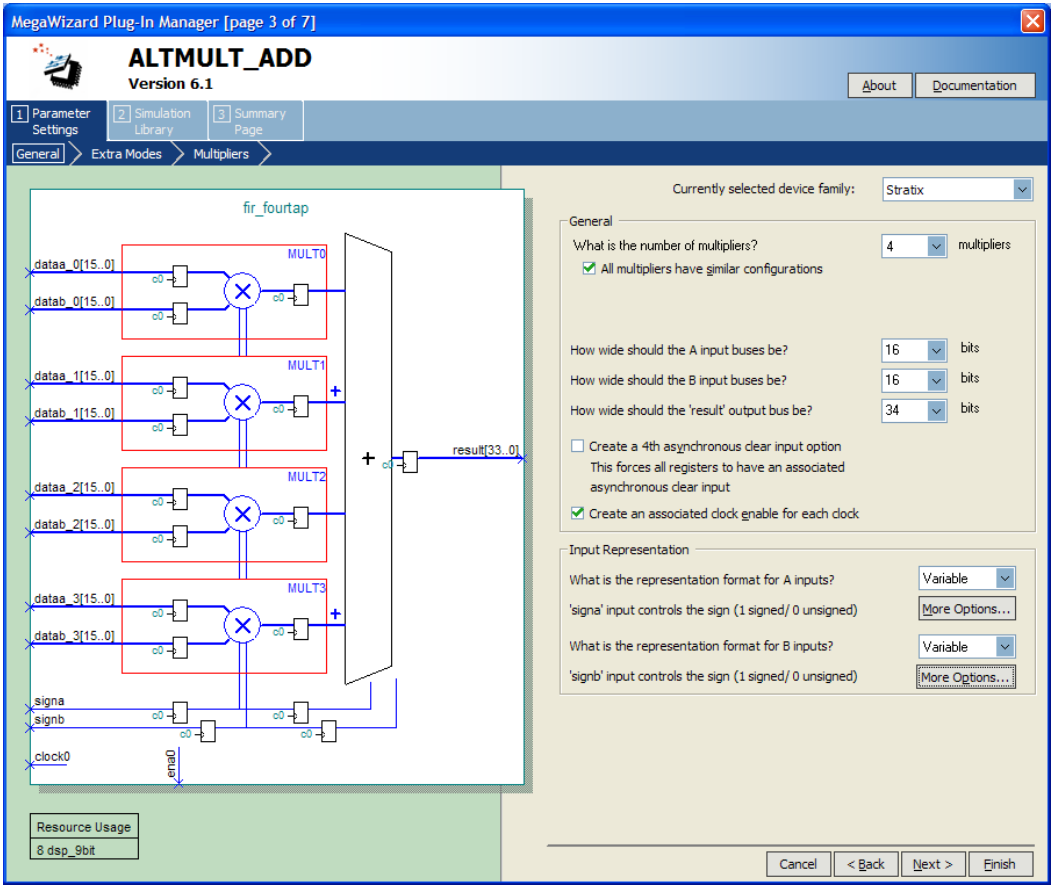
Figure 2–33 shows the **'signa' Register Configuration** dialog box after you have made these selections.

Figure 2–33. 'signb' Register Configuration Dialog Box



24. Click **Done**. Figure 2–34 shows the wizard after you have made these selections.

Figure 2–34. altmult_add Wizard, Page 3



25. Click **Next**.

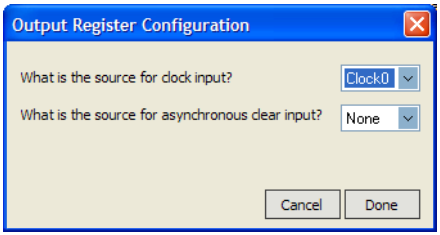
26. Under **Outputs Configuration**, turn off **Create a shiftout output from A input of the last multplier** and **Create a shiftout output from B input of the last multiplier**.

27. Turn on **Register output of the adder unit** and click **More Options**. The **Output Register Configuration** dialog box appears.

28. In the **What is the source for clock input**? list, select **Clock0**.

29. In the **What is the source for asynchronous clear input?** list, select **None**. Figure 2–35 shows the **Output Register Configuration** dialog box after you have made these selections.

Figure 2–35. Output Register Configuration Dialog Box

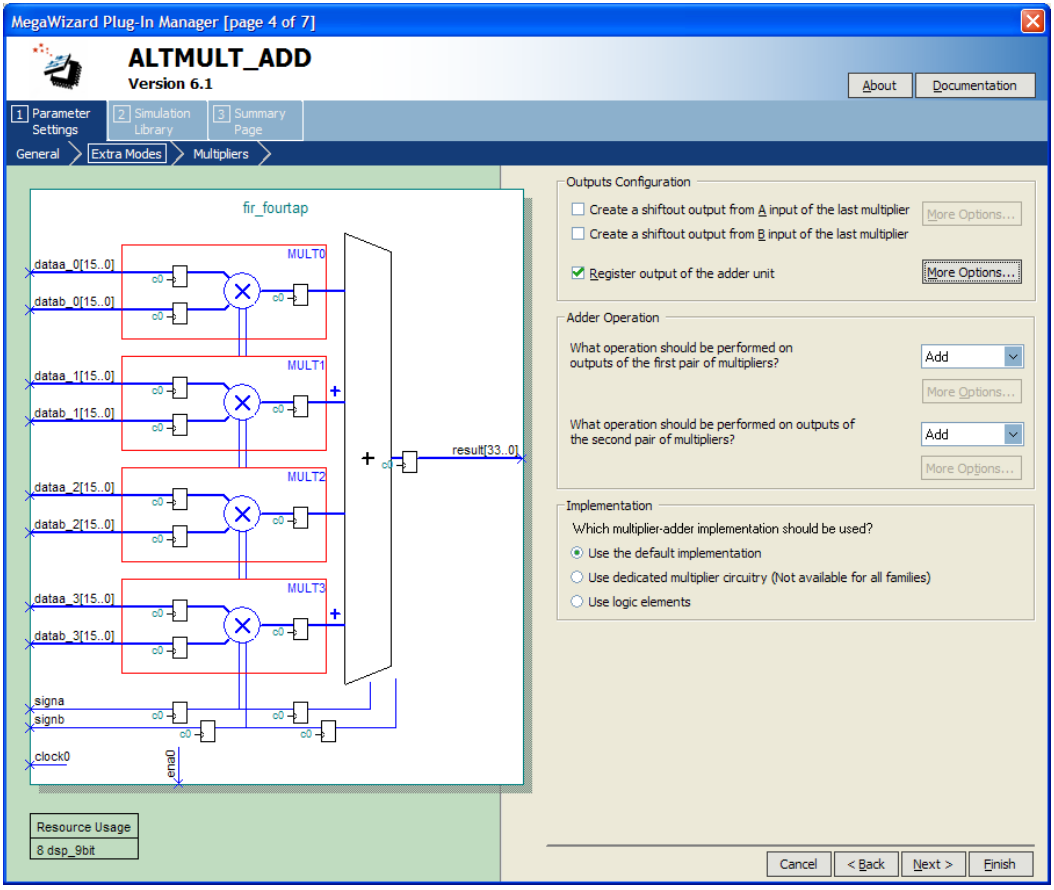


30. Click **Done**.

31. Under **Adder Operation**, in the **What operation should be performed on outputs of the first pair of multipliers?** and the **What operation should be performed on outputs of the second pair of multipliers?** lists, select **Add**.

32. Under **Implementation**, select **Use the default implementation**. Figure 2–36 shows the wizard after you have made these selections.

Figure 2–36. altmult_add Wizard, Page 4



33. Click **Next**.
34. Under **Input Configuration**, turn on **Register input A of the multiplier** and click **More Options**. The **Data A input Register Configuration - Multiplier 0** dialog box appears.
35. In the **What is the source for clock input?** list, select **Clock0**.
36. In the **What is the source for asynchronous clear input?** list, select **None**.
37. Click **Done**.

38. Under **Input Configuration**, turn on **Register input B of the multiplier** and click **More Options**. The **Data B input Register Configuration - Multiplier 0** dialog box appears.

39. In the **What is the source for clock input?** list, select **Clock1**.

40. In the **What is the source for asynchronous clear input?** list, select **None**.

41. Click **Done**.

42. Under **Input Configuration**, in the **What is the input A of the multiplier connected to?** list, select **Shiftin input** and in the **What is the input B of the multiplier connected to?** list, select **Multiplier input**.

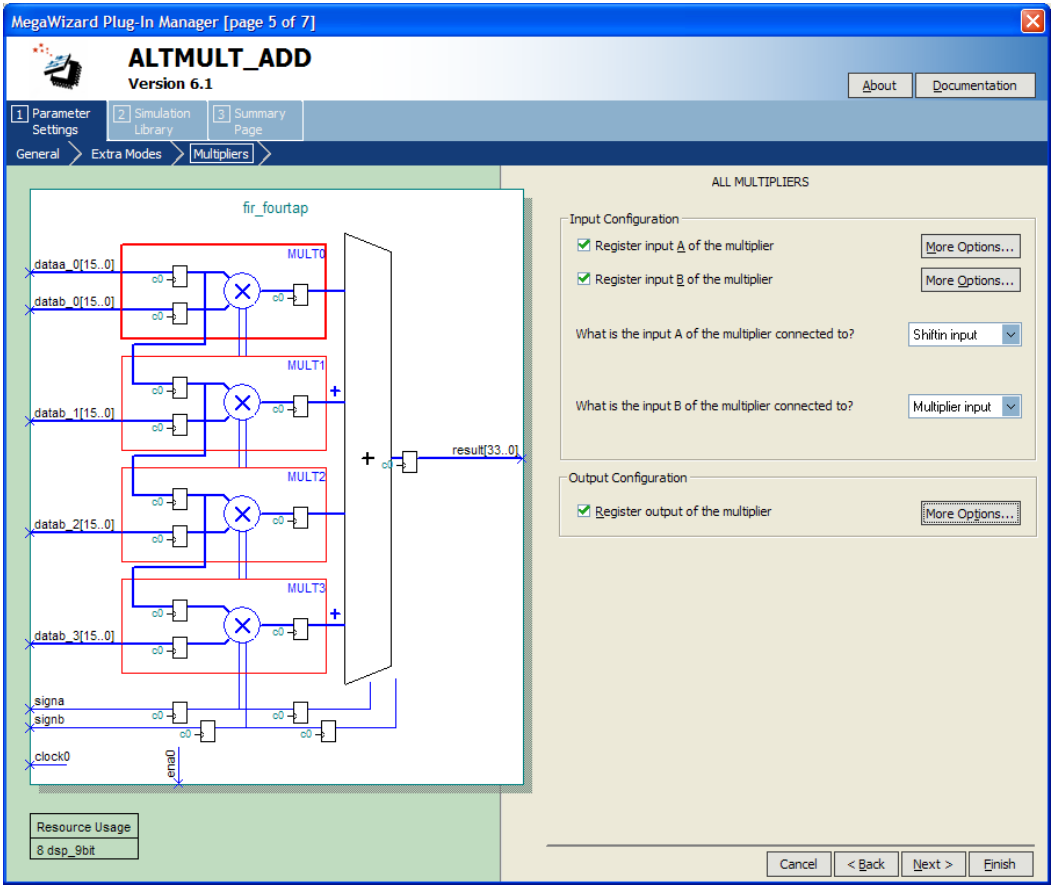
43. Under **Output Configuration**, turn on **Register output of the multiplier** and click **More Options**. The **Output Register Configuration - Multiplier 0** dialog box appears.

44. In the **What is the source for clock input?** list, select **Clock0**.

45. In the **What is the source for asynchronous clear input?** list, select **None**.

46. Click **Done**. Figure 2–37 shows the wizard after you have made these selections.

Figure 2–37. altmult_add Wizard, Page 5



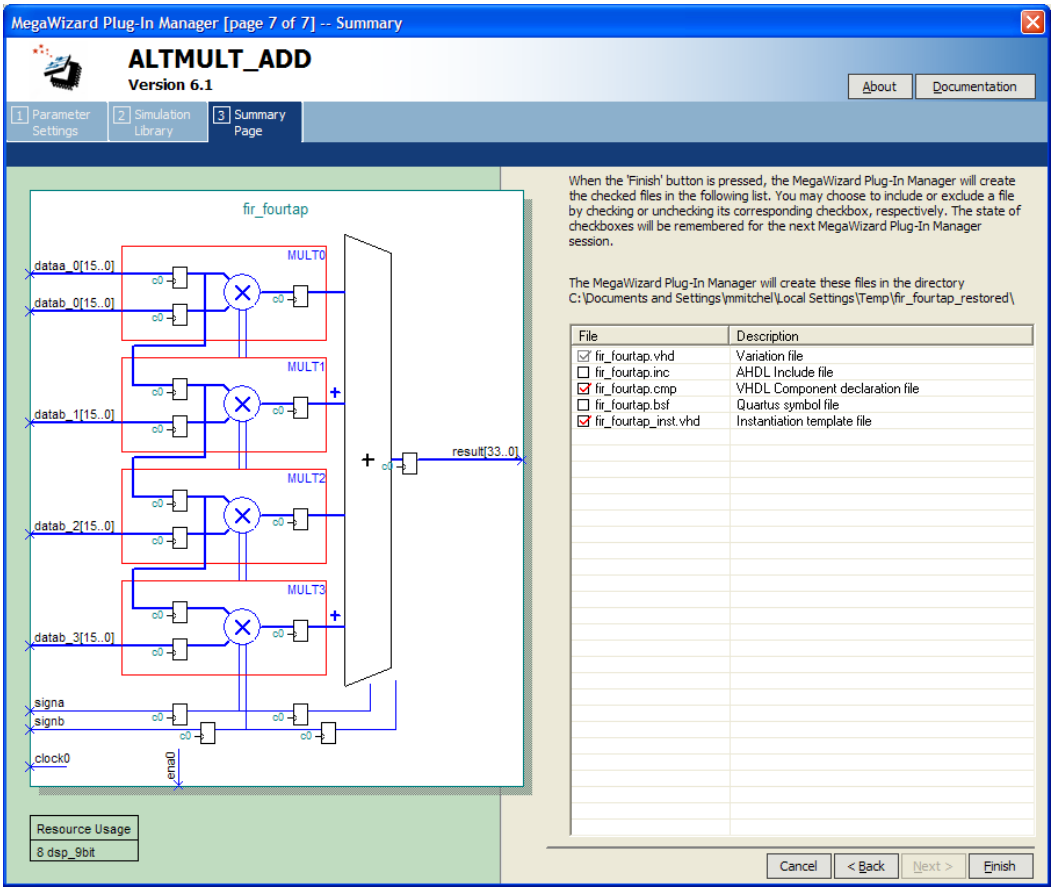
47. Click **Finish**.

The final page of the wizard shows the file that is generated for your custom megafunction variation. The gray check marks indicate files that are always generated; the other files are optional and are generated only if selected (indicated by a red check mark). Turn on the boxes to select the files that you want generated.

48. Turn on **VHDL Component declaration file** and **Instantiation template file**.

49. Turn off **Quartus symbol file** and **AHDL include file**. Leave the other options in their default settings. Figure 2–38 shows the wizard after you have made these selections.

Figure 2–38. altmult_add Wizard, Page 6 Summary



50. Click **Finish**.

Implement the FIR 4-Tap Design

This section describes how to assign the EP1S10F780C5 device to the project and compile the project in the Quartus II software.

1. On the Assignments menu, click **Device**. The **Settings** dialog box appears.

2. In the **Family** list, select **Stratix**.

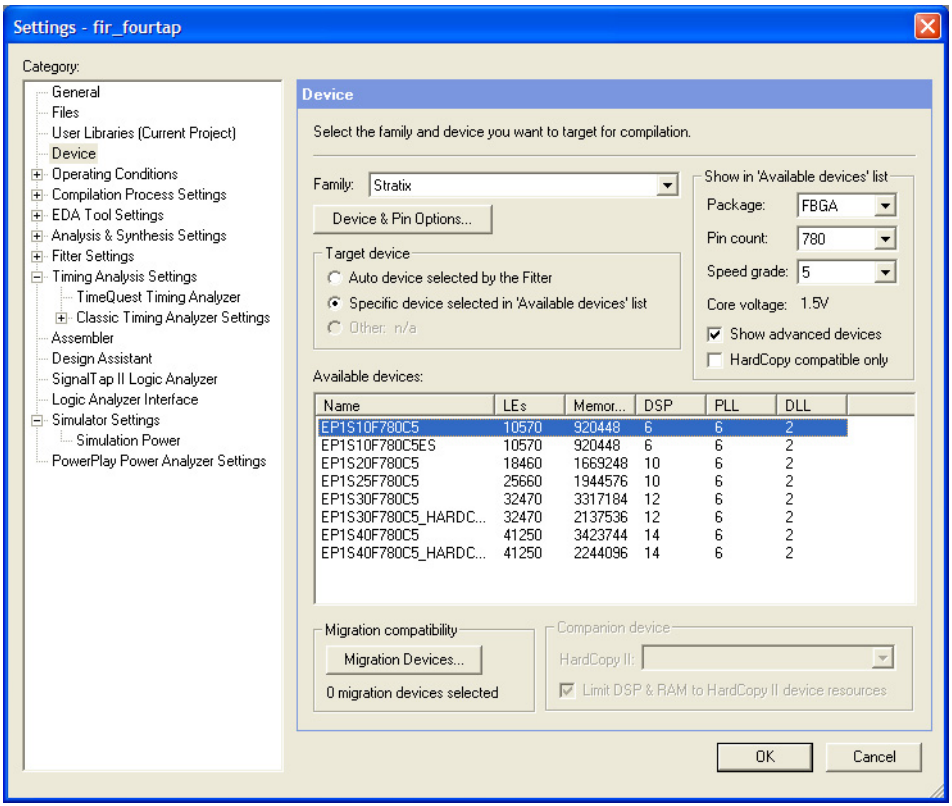
3. Under **Target device**, click **Specific device selected in 'Available devices'** list.

4. Under Show in 'Available devices' list, select the following settings:

   - In the **Package** list, select **FBGA**.
   - In the **Pin count** list, select **780**.
   - In the **Speed grade** list, select **5**.

5. In the **Available devices** list, select **EP1S10F780C5**. Figure 2–39 shows the wizard after you have made these selections.

Figure 2–39. Device Settings



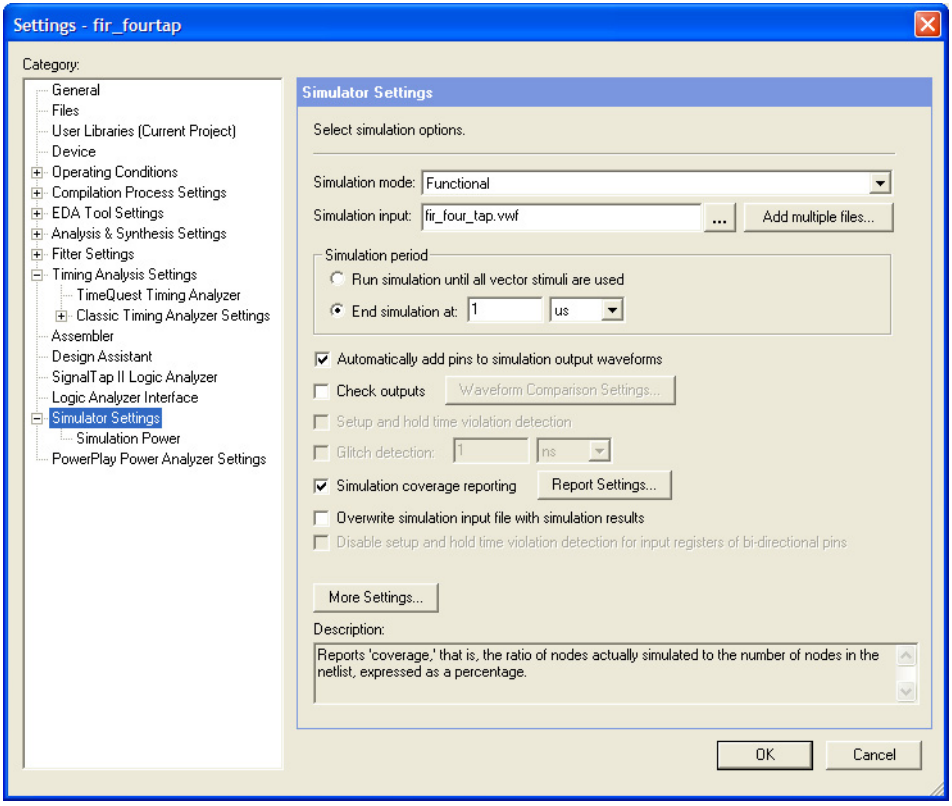
6. Leave the other options in the default state and click **OK**.
7. On the Processing menu, click **Start Compilation**.
8. When the **Full compilation was successful** box appears, click **OK**.

Functional Results—Simulate the FIR 4-Tap Design

This section describes how to verify the design example you just created by simulating the design using the Quartus II Simulator. To set up the Quartus II Simulator, perform the following steps:

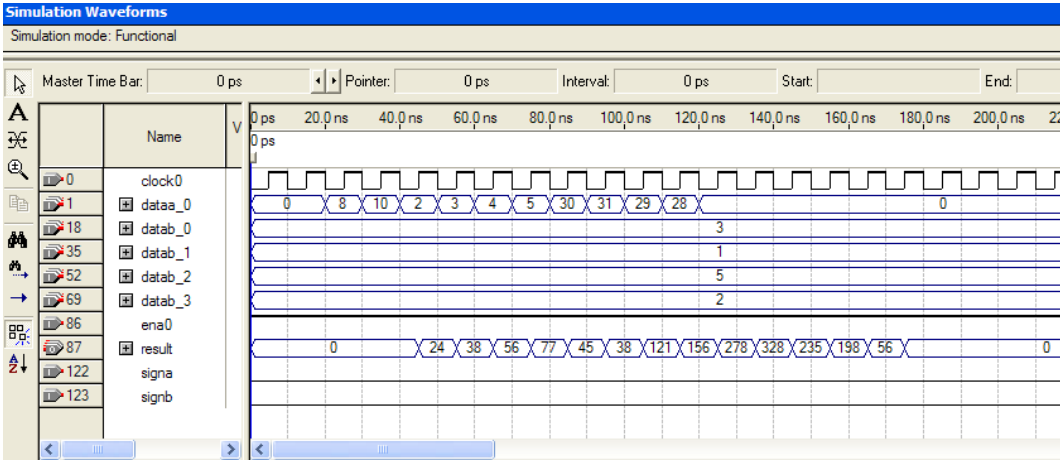
1. On the Processing menu, click **Generate Functional Simulation Netlist**.
2. When the **Functional Simulation Netlist Generation was successful** box appears, click **OK**.
3. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
4. In the **Category** list, select **Simulator Settings**. The **Simulator Settings page** appears.
5. In the **Simulation mode** list, select **Functional**.
6. Type `fir_four_tap.vwf` in the **Simulation Input** box, or click **Browse** (**...**) to select the file in the project folder.
7. Turn on **End simulation at:**, type `1`, and select **us**.
8. Turn on **Automatically add pins to simulation output waveforms** and **Simulation coverage reporting**.
9. Turn off **Check outputs** and **Overwrite simulation input file with simulation results**. Figure 2–40 shows the **Simulator Settings** page after you have made these selections.

Figure 2–40. Simulator Settings Page



10. In the Category list, click the + icon to expand **Simulator Settings** and select **Simulator Power**. The **Simulator Power** page appears.
11. Turn off **Generate Signal Activity File** and **Generate VCD File**.
12. Click **OK**.
13. On the Processing menu, click **Start Simulation** to run a simulation.
14. When the **Simulation was successful** box appears, click **OK**.
15. The Simulation Report window appears. Verify the simulation waveform results (Figure 2–41).

Figure 2–41. Functional Simulation Waveform

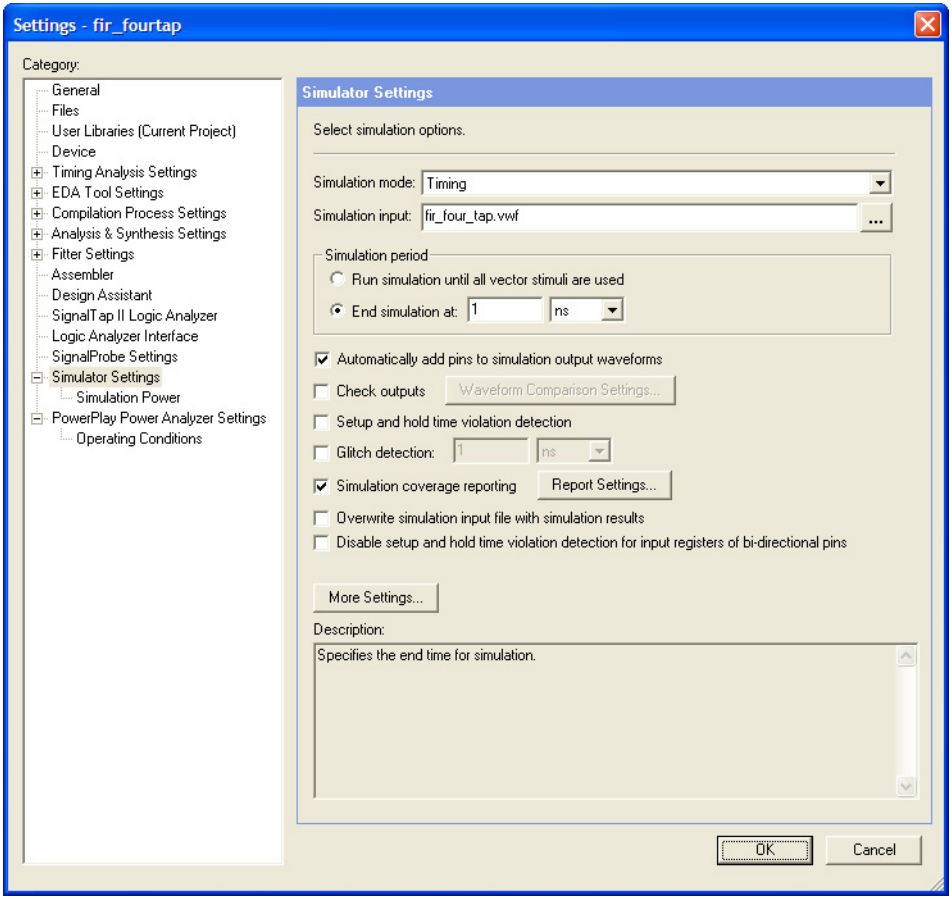


Timing Results

This section describes how to verify the timing after implementation using the Quartus II Simulator. To set up the Quartus II Simulator, perform the following steps:

1. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
2. In the **Category** list, select **Simulator Settings**.
3. In the **Simulation mode** list, select **Timing**.
4. In the **Simulation Input** box, type `fir_four_tap.vwf`, or click **Browse (...)** to select the file in the project folder.
5. Turn on the **End simulation at:,** type `1`, and select **us**.
6. Turn on **Automatically add pins to simulation output waveforms** and **Simulation coverage reporting**.
7. Turn off **Check outputs**, **Setup and hold time violation detection**, **Glitch detection**, and **Overwrite simulation input file with simulation results**. Figure 2–42 shows the **Simulator Settings** page after you have made these selections.

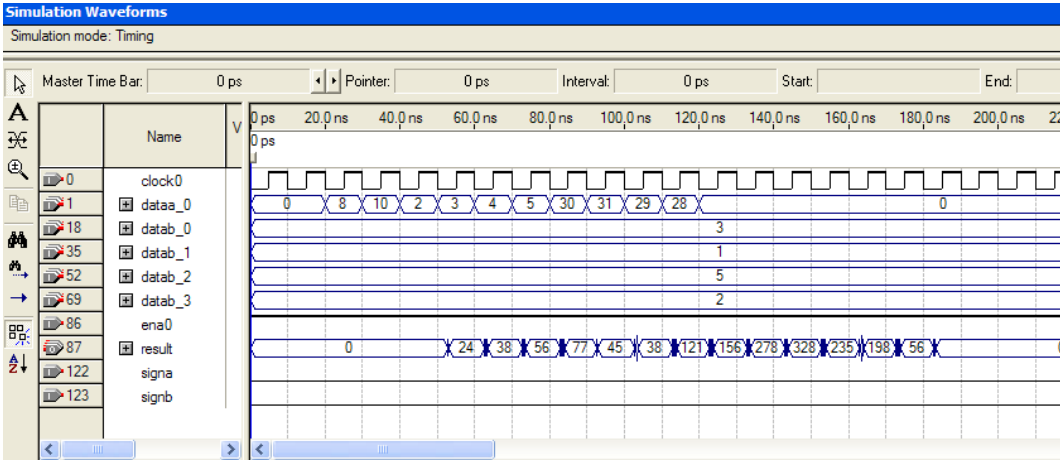
Figure 2–42. Simulator Settings Page



8. In the Category list, click the + icon to expand **Simulator Settings** and select **Simulator Power**. The **Simulator Power** page appears.
9. Turn off **Generate Signal Activity File** and **Generate VCD File**.
10. Click **OK**.
11. On the Processing menu, click **Start Simulation** to run a simulation. You may be prompted to save the file; if so, click **OK**.
12. When the **Simulation was successful** box appears, click **OK**.

13. The Simulation Report window appears. Verify the simulation waveform results (Figure 2–43).

Figure 2–43. Timing Simulation Waveform



Simulate the FIR Filter Design in the ModelSim-Altera Software

Simulate the design in ModelSim to compare the results of both simulators. Setup the ModelSim-Altera simulator by performing the following steps:

1. Unzip **almult_add_ex2_msim.zip** to any working directory on your PC.

2. Start ModelSim-Altera.

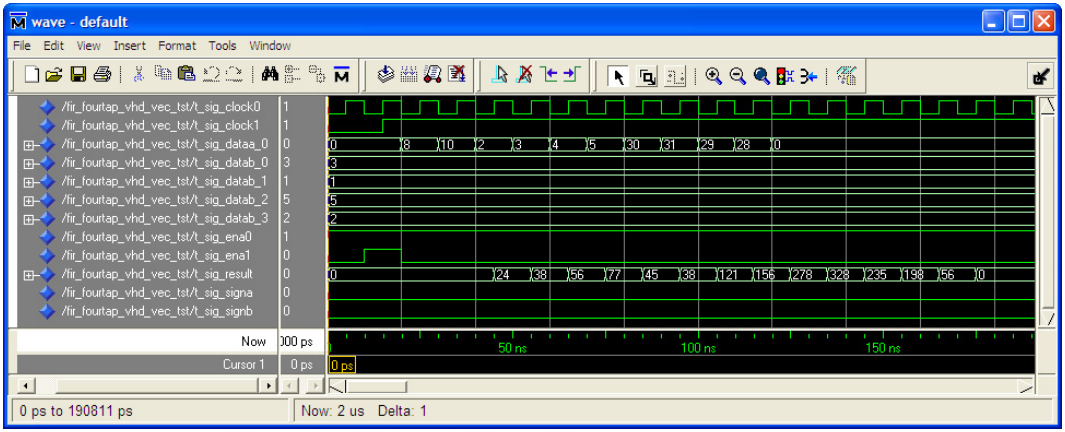
3. On the File menu, click **Change Directory**. The **Choose folder** dialog box appears in whcih to select the folder where you have unzipped your files. Click **OK**.

4. On the Tools menu, click **Execute Macro**. The **Execute Do File** dialog box appears.

5. Select **fir_fourtap_functional.do** and click **Open**.

fir_fourtap_functional.do is a script file for ModelSim that automates all the necessary settings for the functional simulation. You can verify the results in the Waveform Viewer window (Figure 2–44).

Figure 2–44. ModelSim Waveform Viewer for Functional Simulation



If needed, you can rearrange signals, remove redundant signals, and change the radix to suit the results in the Quartus II Simulator.

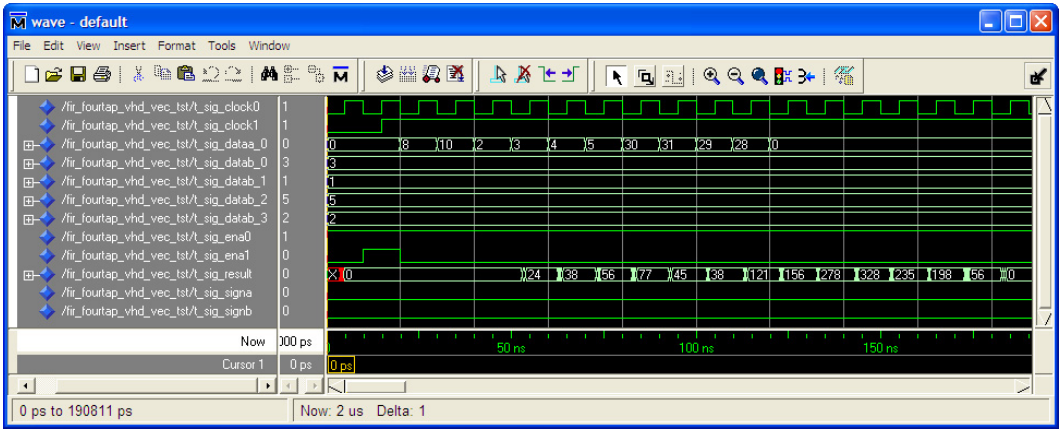
To further check the timing simulation, perform the following steps:

1. In the Tools menu, click **Execute Macro**. The **Execute Do File** dialog box appears.

2. Select **fir_fourtap_timing.do** and click **Open**.

fir_fourtap_timing.do is a script file for ModelSim that automates all the necessary settings for the timing simulation. You can verify the results in the Waveform Viewer window (Figure 2–45).

Figure 2–45. ModelSim Waveform Viewer for Timing Simulation



If needed, you can rearrange signals, remove redundant signals, and change the radix to suit the results in the Quartus II Simulator.

Conclusion

The Quartus II software provides parameterizable megafunctions ranging from simple arithmetic units, such as adders and counters, to advanced phase-locked loop (PLL) blocks, multipliers, and memory structures. These megafunctions are performance-optimized for Altera devices and therefore provide more efficient logic synthesis and device implementation because they automate the coding process and save valuable design time. Altera recommends using these functions during design implementation so you can consistently meet your design goals.

Chapter 3. Specifications

Ports and Parameters

The options listed in this section describe all of the ports and parameters that are available for each device to customize the altmult_add megafunction according to your application.

Figure 3–1 shows the ports and parameters for the altmult_add megafunction.

Figure 3–1. Port and Parameter Description

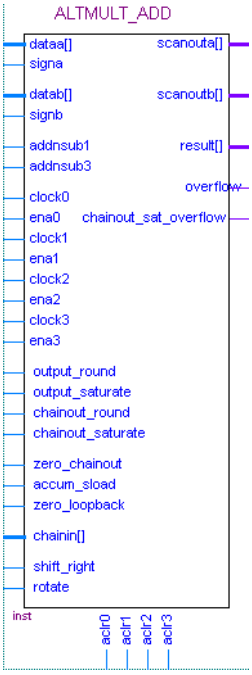


Table 3–1 shows the input ports, Table 3–2 shows the output ports, and Table 3–3 shows the parameters for the altmult_add megafunction.

The parameter details are only relevant for users who by-pass the MegaWizard® Plug-In Manager interface and use the megafunction as a directly parameterized instantiation in their design. The details of these parameters are hidden from the user of the MegaWizard Plug-In Manager interface.

Refer to the latest version of the Quartus® II Help for the most current information on the ports and parameters for this megafunction.

LOOPBACK mode is only supported when each input is 18 bits wide, width_result is 36 bits wide, and the number of multipliers is either 1 or 2.

Table 3–1. altmult_add Megafunction Input Ports (Part 1 of 3)

Port Name	Required	Description	Comments
dataa[]	Yes	Data input to the multiplier.	Input port [NUMBER_OF_MULTIPLIERS * WIDTH_A - 1..0] wide.
datab[]	Yes	Data input to the multiplier.	Input port [NUMBER_OF_MULTIPLIERS * WIDTH_B - 1..0] wide.
clock[]	No	Clock input, usable by any register in the megafunction.	Input port [0..3]. Clock input to the corresponding register.
aclr[]	No	Asynchronous clear input.	Input port [0..3]. Asynchronous clear input to the corresponding register.
ena[]	No	Clock enable for the clock[] port.	Input port [0..3]. Clock enable for the corresponding clock[] port.
output_round	No	Enables dynamically controlled output rounding.	When OUTPUT_ROUNDING is set to VARIABLE, output_round enables the final adder stage of rounding. *(1)*
output_saturate	No	Enables dynamically controlled output saturation.	When OUTPUT_SATURATION is set to VARIABLE, output_saturate enables the final adder stage of saturation. *(1)*
chainout_round	No	Enables dynamically controlled chainout stage rounding.	When CHAINOUT_ROUNDING is set to VARIABLE, chainout_round enables the chainout stage of rounding. *(1)*
chainout_saturate	No	Enables dynamically controlled chainout stage saturation.	When CHAINOUT_SATURATION is set to VARIABLE, chainout_saturate enables the chainout stage of saturation. *(1)*
zero_chainout	No	Dynamically specifies whether the chainout value is zero.	*(1)*

Table 3–1. altmult_add Megafunction Input Ports (Part 2 of 3)

Port Name	Required	Description	Comments
zero_loopback	No	Dynamically specifies whether the loopback value is zero.	*(1)*
accum_sload	No	Dynamically specifies whether the accumulator value is zero.	*(1)*
chainin	No	Adder result input bus from the preceding stage.	Input port [WIDTH_CHAININ - 1..0] wide. *(1)*
rotate	No	Specifies dynamically controlled port rotation in shift mode.	*(1)*
shift_right	No	Specifies dynamically controlled port shift right or left in shift mode.	Values are 0 and 1. A value of 0 specifies a shift to the left, a value of 1 specifies a shift to the right. *(1)*
signa	No	Specifies the numerical representation of the dataa[] port.	If the signa port is high, the multiplier treats the dataa[] port as a signed two's complement number. If the signa port is low, the multiplier treats the dataa[] port as an unsigned number.
signb	No	Specifies the numerical representation of the datab[] port.	If the signb port is high, the multiplier treats the datab[] port as a signed two's complement number. If the signb port is low, the multiplier treats the datab[] port as an unsigned number.
scanina[]	No	Input for scan chain A.	Input port [WIDTH_A - 1..0] wide. When the INPUT_SOURCE_A parameter has a value of SCANA or VARIABLE, the scanina port is required. Do not use scanina[] and scaninb[] simultaneously. *(2)*
scaninb[]	No	Input for scan chain B.	Input port [WIDTH_B - 1..0] wide. When the INPUT_SOURCE_A parameter has a value of SCANB or VARIABLE, the scaninb port is required. Do not use scanina[] and scaninb[] simultaneously. *(2)*
sourcea	No	Input source for scan chain A.	*(2)*
sourceb	No	Input source for scan chain B.	*(2)*

Table 3–1. altmult_add Megafunction Input Ports (Part 3 of 3)

Port Name	Required	Description	Comments
addnsub1	No	Controls the functionality of the adder.	If the addnsub1 port is high, the adder performs an add function. If the addnsub1 port is low, the adder performs a subtract function. *(2)*
addnsub1_round	No	Enables adding or subtracting for the second multiplier.	*(2)*
addnsub3	No	Controls the functionality of the adder	If the addnsub3 port is high, the adder performs an add function. If the addnsub3 port is low, the adder performs a subtract function. *(2)*
addnsub3_round	No	Enables adding or subtracting for the fourth multiplier.	*(2)*
mult[]_round	No	Enables rounding for the first and second multiplier [01], or the third and fourth multiplier [23].	Port [01,23]. Port is required when the corresponding MULTIPLIER[]_ROUNDING parameter has a value of VARIABLE. *(2)*
mult[]_saturation	No	Enables saturation for the first and second multiplier [01], or the third and fourth multiplier [23].	Port [01,23]. Port is required when the corresponding MULTIPLIER[]_SATURATION parameter has a value of VARIABLE. *(2)*

Notes to Table 3–1:
(1) This parameter is available only for Stratix III devices.
(2) This parameter is available only for Stratix II devices.

Table 3–2. altmult_add Megafunction Output Ports

Port Name	Required	Description	Comments
result[]	Yes	Multiplier output port	Output port [WIDTH_RESULT – 1..0] wide.
overflow	No	Overflow flag	If output_saturation is enabled, overflow flag is set.
chainout_sat_overflow	No	Overflow flag for the chainout saturation	*(1)*

scanouta[]	No	Output of scan chain A	Output port [WIDTH_A - 1..0] wide. When designing with Stratix III devices, port cannot be selected when scaninb is in use. Note: Do not use scanina[] and scaninb[] simultaneously.
scanoutb[]	No	Output of scan chain B	Output port [WIDTH_B - 1..0] wide. When designing with Stratix III devices, port cannot be selected when scanina is in use. Note: Do not use scanina[] and scaninb[] simultaneously.
mult0_is_saturated	No	Signal indicating saturation of the first multiplier	This port is required when PORT_MULT0_IS_SATURATED has a value of USED. *(2)*
mult1_is_saturated	No	Signal indicating saturation of the second multiplier	This port is required when PORT_MULT1_IS_SATURATED has a value of USED. *(2)*
mult2_is_saturated	No	Signal indicating saturation of the third multiplier	This port is required when PORT_MULT2_IS_SATURATED has a value of USED. *(2)*
mult3_is_saturated	No	Signal indicating saturation of the fourth multiplier	this port is required when PORT_MULT3_IS_SATURATED has a value of USED. *(2)*

Note to Table 3–2:

(1) This parameter is available only for Stratix® III devices.

(2) This parameter is available only for Stratix II devices.

Table 3–3. altmult_add Megafunction Parameters (Part 1 of 18)

Parameter	Type	Required	Comments
NUMBER_OF_MULTIPLIERS	String	Yes	Number of multipliers to be added together. Values are 1 up to 4.
WIDTH_A	String	Yes	Width of the dataa[] port
WIDTH_B	String	Yes	Width of the datab[] port
WIDTH_RESULT	String	Yes	Width of the result[] port. Value includes all bits before rounding and saturation.
WIDTH_CHAININ	Integer	No	Width of the chainin port. WIDTH_CHAININ equals WIDTH_RESULT if port chainin is used. If omitted, the default value is 1. *(1)*

Table 3–3. altmult_add Megafunction Parameters (Part 2 of 18)

Parameter	Type	Required	Comments
INPUT_REGISTER_A[]	String	No	Parameter [0..3]. Specifies the clock port for the dataa[] operand of the first multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default is CLOCK0. For Stratix III devices, values [1..3] should follow INPUT_REGISTER_A0.
INPUT_ACLR_A[]	String	No	Parameter [0..3]. Specifies the asynchronous clear for the dataa[] operand of the first multiplier. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and corresponding INPUT_REGISTER_A[] is used, the default is ACLR3. Values [1..3] should follow INPUT_ACLR_A0.
INPUT_SOURCE_A[]	String	No	Parameter [0..3]. Specifies the data source to the first multiplier. Values are DATAA and SCANA. If this parameter is set to DATAA, the adder uses the values from the dataa[] port. If this parameter is set to SCANA, the adder uses values from the scan chain. If omitted, the default is DATAA. For Stratix II devices, a value of VARIABLE is also available.
REPRESENTATION_A	String	No	Specifies the numerical representation of the dataa[] port. Values are UNSIGNED and SIGNED. When this parameter is set to UNSIGNED, the adder interprets the dataa[] input as an unsigned number. When this parameter is set to SIGNED, the adder interprets the dataa[] input as a signed two's complement number. If the signa port is used, this parameter is ignored. If omitted, the default is UNSIGNED.
REPRESENTATIONS_B	String	No	Specifies the numerical representation of the datab[] port. Values are UNSIGNED and SIGNED. When this parameter is set to UNSIGNED, the adder interprets the datab[] input as an unsigned number. When this parameter is set to SIGNED, the adder interprets the datab[] input as a signed two's complement number. If the signb port is used, this parameter is ignored. If omitted, the default is UNSIGNED.

Table 3–3. altmult_add Megafunction Parameters (Part 3 of 18)

Parameter	Type	Required	Comments
SIGNED_REGISTER_[]	String	No	Parameter [A,B]. Specifies the clock signal for the first register on the corresponding sign[] port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If the corresponding sign[] port value is UNUSED, this parameter is ignored. If omitted, the default is CLOCK0.
SIGNED_ACLR_[]	String	No	Parameter [A,B]. Specifies the asynchronous clear signal for the first register on the corresponding sign[] port. Values are NONE, ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and corresponding SIGNED_REGISTER_[] is used, the default is ACLR3.
SIGNED_PIPELINE_ REGISTER_[]	String	No	Parameter [A,B]. Specifies the clock signal for the second register on the corresponding sign[] port. Values are UNREGISTERED, CLOCK0, CLOCK1,CLOCK2, and CLOCK3. If the corresponding sign[] port value is UNUSED, this parameter is ignored. If omitted, the default is CLOCK0.
SIGNED_PIPELINE_ACLR_[]	String	No	Parameter [A,B]. Specifies the asynchronous clear signal for the second register on the corresponding sign[] port. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and the corresponding SIGNED_PIPELINE_REGISTER_[] is used, the default is ACLR3.
SCANOUTA_REGISTER	String	No	Specifies the clock source for the scanouta data bus registers. UNREGISTERED, CLOCK0, CLOCK1,CLOCK2, and CLOCK3. If omitted, the default is UNREGISTERED. *(1)*
SCANOUTA_ACLR	String	No	Specifies the asynchronous clear source for the scanouta data bus registers. Legal values are ACLR0,ACLR1, ACLR2, and ACLR3. If omitted and SCANOUTA_REGISTER is used, the default is ACLR3. *(1)*

Table 3–3. altmult_add Megafunction Parameters (Part 4 of 18)

Parameter	Type	Required	Comments
INPUT_REGISTER_B[]	String	No	Parameter [0..3]. Specifies the clock port for the datab[] operand of the corresponding multiplier. Values are UNREGISTERED, CLOCK0, CLOCK1,CLOCK2, and CLOCK3. If omitted, the default is CLOCK0.
INPUT_ACLR_B[]	String	No	Parameter [0..3]. Specifies the asynchronous clear for the datab[] operand of the corresponding multiplier. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and corresponding INPUT_REGISTER_B[] is used, the default is ACLR3.
INPUT_SOURCE_B[]	String	No	Parameter [0..3]. Specifies the data source of the corresponding multiplier. Values are DATAB, SCANB, and VARIABLE. If this parameter is set to DATAB, then the adder uses the values from the datab[] port. If this parameter is set to SCANB, then the adder uses values from the scan chain. If omitted, the default is DATAB. Stratix devices support value DATAB only. Stratix II devices support values DATAB, SCANB and VARIABLE only. Stratix III devices supports values DATAB, SCANB and LOOPBACK. LOOPBACK value is in sum2 mode.
MULTIPLIER_REGISTER[]	String	No	Parameter [0..3]. Specifies the clock source for the register immediately following the corresponding multiplier. Values are UNREGISTERED, CLOCK0,CLOCK1, CLOCK2, and CLOCK3. If omitted, the default is CLOCK0.
MULTIPLIER_ACLR[]	String	No	Parameter [0..3]. Specifies the asynchronous clear signal for the register immediately following the corresponding multiplier. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and corresponding MULTIPLIER_REGISTER[] is used, the default is ACLR3.

Table 3–3. altmult_add Megafunction Parameters (Part 5 of 18)

Parameter	Type	Required	Comments
MUTIPLIER1_DIRECTION	String	No	Specifies whether the second multiplier adds or subtracts its value from the sum. Values are ADD and SUB. If the addnsub1 port is used, this parameter is ignored. If omitted, the default is ADD.
MUTIPLIER3_DIRECTION	String	No	Specifies whether the fourth and all subsequent odd-numbered multipliers add or subtract their results from the total. Values are ADD and SUB. If the addnsub3 port is used, this parameter is ignored. If omitted, the default is ADD.
ACCUM_DIRECTION	String	No	Specifies whether to use the accumulator and whether the accumulator adds or subtracts its value from the sum. Values are ADD and SUB. If omitted, the default is ADD.
CHAINOUT_REGISTER	String	No	Specifies the clock source for the chainout mode result register. This is an additional stage after the second adder. Values are UNREGISTERED, CLOCK0,CLOCK1, CLOCK2, and CLOCK3. If omitted, the default is CLOCK0. *(1)*
CHAINOUT_ACLR	String	No	Specifies the asynchronous clear for the chainout mode result register. This is an additional stage after the second adder. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and CHAINOUT_REGISTER is used, the default is ACLR3. *(1)*
ADDNSUB_MULTIPLIER_ REGISTER[]	String	No	Parameter [1,3]. Specifies the clock signal for the first register on the corresponding addnsub[] input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If the corresponding addnsub[] port is UNUSED, this parameter is ignored. If omitted, the default is CLOCK0. *(2)*

Table 3–3. altmult_add Megafunction Parameters (Part 6 of 18)

Parameter	Type	Required	Comments
ADDSUB_MULTIPLIER_ACLR[]	String	No	Parameter [1,3]. Specifies the asynchronous clear signal for the first register on the corresponding addnsub[] input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If the corresponding addnsub[] port value is UNUSED, this parameter is ignored. If omitted and corresponding ADDNSUB_MULTIPLIER_REGISTER[] is used, the default is ACLR3. *(2)*
ADDNSUB_MULTIPLIER_PIPELINE_REGISTER[]	String	No	Parameter [1,3]. Specifies the clock signal for the second register on the corresponding addnsub[] input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If the corresponding addnsub[] port is UNUSED, this parameter is ignored. If omitted, the default is CLOCK0. *(2)*
ADDNSUB_MULTIPLIER_PIPELINE_ACLR[]	String	No	Parameter [1,3]. Specifies the asynchronous clear signal for the second register on the corresponding addnsub[] input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and corresponding ADDNSUB_MULTIPLIER_PIPELINE_REGISTER[] is used, the default is ACLR3.*(2)*
OUTPUT_REGISTER	String	No	Specifies the clock signal for the second adder register. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default is CLOCK0.
OUTPUT_ACLR	String	No	Specifies the asynchronous clear signal for the second adder register. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted, the default is ACLR3.
PORT_ADDNSUB[]	String	No	Parameter [1,3]. Specifies the usage of the corresponding addnsub[] input port. Values are PORT_USED, PORT_UNUSED, and PORT_CONNECTIVITY. A value of PORT_CONNECTIVITY specifies the port usage by checking port connectivity. If omitted, the default is PORT_CONNECTIVITY. *(2)*

Table 3–3. altmult_add Megafunction Parameters (Part 7 of 18)

Parameter	Type	Required	Comments
MULTIPLIER[]_ROUNDING	String	No	Parameter [01,23]. Specifies rounding for the first and second multiplier [01], or the third and fourth multiplier [23]. Values are NO, YES, and VARIABLE. If omitted, the default is NO. *(2)*
MULTIPLIER[]_SATURATION	String	No	Parameter [01,23]. Specifies saturation for the first and second multiplier [01], or the third and fourth multiplier [23]. Values are NO, YES, and VARIABLE. If omitted, the default is NO. *(2)*
ADDER[]_ROUNDING	String	No	Parameter [1,3]. Specifies adder rounding for the first multiplier [1], or the third multiplier [3]. Values are NO, YES, and VARIABLE. If omitted, the default is NO. *(2)*
PORT_MULT[]_IS_SATURATED	String	No	Parameter [0..3]. Specifies whether to use the corresponding mult[]_is_saturated output port. Values are NO and YES. If omitted, the default is NO. *(2)*
PORT_SIGN[]	String	No	Parameter [A,B]. Specifies the corresponding sign[] input port usage. Values are PORT_USED, PORT_UNUSED, and PORT_CONNECTIVITY. If omitted, the default is PORT_CONNECTIVITY.
OUTPUT_ROUNDING	String	No	Enables rounding handling at second adder stage. If original design uses a Stratix II device, in some cases this parameter can be derived from the Stratix II rounding settings. Values are YES, NO, and VARIABLE. A value of YES or NO specifies saturation handling setting permanently to on or off. A value of VARIABLE allows dynamically controlled saturation handling. *(1)*
OUTPUT_ROUND_TYPE	String	No	Specifies the rounding mode. Values are NEAREST_EVEN and NEAREST_INTEGER. • A value of NEAREST_EVEN specifies round-to-nearest-even. • A value of NEAREST_INTEGER specifies round-to-nearest-integer. If omitted, the default is NEAREST_INTEGER. *(1)*

Table 3–3. altmult_add Megafunction Parameters (Part 8 of 18)

Parameter	Type	Required	Comments
WIDTH_MSB	Integer	No	Specifies the fractional rounding width. The value is determined by counting the bits from the MSB (before saturation) to the LSB (after rounding). Values are calculated according to the following modes: WIDTH_A, WIDTH_B, and WIDTH_RESULT. Value must be an unsigned integer, and must be less than WIDTH_RESULT. If a positive number is unavailable, no saturation is allowed in your input/output width and mode setting. If omitted, the default value is 17, which is compatible with Stratix II device settings. *(2)*
OUTPUT_ROUND_REGISTER	String	No	Specifies the clock source for the first register on the output_round input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*
OUTPUT_ROUND_ACLR	String	No	Specifies the asynchronous clear source for the first register on the output_round input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and OUTPUT_ROUND_REGISTER is used, the default value is ACLR3. *(1)*
OUTPUT_ROUND_PIPELINE_REGISTER	String	No	Specifies the clock source for the second register on the output_round input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*
OUTPUT_ROUND_PIPELINE_ACLR	String	No	Specifies the asynchronous clear source for the second register on the output_round input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and OUTPUT_ROUND_PIPELINE_REGISTER is used, the default value is ACLR3. *(1)*

Table 3–3. altmult_add Megafunction Parameters (Part 9 of 18)

Parameter	Type	Required	Comments
OUTPUT_SATURATION	String	No	Enables saturation handling at second adder stage. If original design uses a Stratix II device, in some cases this parameter can be derived from the Stratix II rounding settings. Values are YES, NO, and VARIABLE. A value of YES or NO specifies saturation handling setting permanently to on or off. A value of VARIABLE allows dynamically controlled saturation handling. If omitted, the default value is NO. *(1)*
OUTPUT_SATURATE_TYPE	String	No	Specifies the saturation mode. Values are SYMMETRIC and ASYMMETRIC. A value of SYMMETRIC specifies the absolute value of the maximum negative number equal to the maximum positive number. A value of ASYMMETRIC specifies the maximum negative number as large as the maximum positive number. If omitted, the default value is ASYMMETRIC. *(3)*
WIDTH_SATURATE_SIGN	String	No	Specifies the saturation position. The value is determined by counting the bits that become the sign bits after saturation. Values are calculated according to the following modes: WIDTH_A, WIDTH_B, and WIDTH_RESULT. Value must be an unsigned integer. If a positive number is unavailable, no saturation is allowed in your input/output width and mode setting. If omitted, the default value is 1. *(3)*
OUTPUT_SATURATE_ REGISTER	String	No	Specifies the clock source for the first register on the output_saturate input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is UNREGISTERED. *(1)*
OUTPUT_SATURATE_ACLR	String	No	Specifies the asynchronous clear source for the first register on the output_saturate input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and OUTPUT_SATURATE_ REGISTER is used, the default value is ACLR3. *(1)*

Table 3–3. altmult_add Megafunction Parameters (Part 10 of 18)

Parameter	Type	Required	Comments
OUTPUT_SATURATE_PIPELINE_REGISTER	String	No	Specifies the clock source for the second register on the output_saturate input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*
OUTPUT_SATURATE_PIPELINE_ACLR	String	No	Specifies the asynchronous clear source for the second register on the output_saturate input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and OUTPUT_SATURATE_PIPELINE_REGISTER is used, the default value is ACLR3.*(1)*
CHAINOUT_ROUNDING	String	No	Enables rounding handling at the chainout stage. Values are YES, NO, and VARIABLE. A value of YES or NO specifies saturation handling setting permanently to on or off. A value of VARIABLE allows dynamically controlled saturation handling. If omitted, the default value is NO. *(1)*
CHAINOUT_ROUND_TYPE	String	No	Specifies the rounding mode at the chainout stage. Values are BIASED and UNBIASED. A value of BIASED specifies round-to-nearest-integer. A value of UNBIASED specifies round-to-nearest-even.
CHAINOUT_ROUND_FRACTION_WIDTH	Integer	No	Specifies the fractional rounding width at the chainout stage. Values are [0..15].
CHAINOUT_ROUND_REGISTER	String	No	Specifies the clock source for the first register on the chainout_round input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*
CHAINOUT_ROUND_ACLR	String	No	Specifies the asynchronous clear source for the first register on the chainout_round input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and CHAINOUT_ROUND_REGISTER is used, the default value is ACLR3. *(1)*
CHAINOUT_ROUND_PIPELINE_REGISTER	String	No	Specifies the clock source for the second register on the chainout_round input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*

Table 3–3. altmult_add Megafunction Parameters (Part 11 of 18)

Parameter	Type	Required	Comments
CHAINOUT_ROUND_PIPELINE_ACLR	String	No	Specifies the asynchronous clear source for the second register on the chainout_round input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and CHAINOUT_ROUND_PIPELINE_REGISTER is used, the default value is ACLR3. *(1)*
CHAINOUT_ROUND_OUTPUT_REGISTER	String	No	Specifies the clock source for the third register on the chainout_round input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*
CHAINOUT_ROUND_OUTPUT_ACLR	String	No	Specifies the asynchronous clear source for the third register on the chainout_round input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and CHAINOUT_ROUND_OUTPUT_REGISTER is used, the default value is ACLR3. *(1)*
CHAINOUT_SATURATION	String	No	Enables saturation handling at the chainout stage. Values are YES, NO, and VARIABLE. A value of YES or NO specifies saturation handling setting permanently to on or off. A value of VARIABLE allows dynamically controlled saturation handling. If omitted, the default value is NO. *(1)*
CHAINOUT_SATURATE_REGISTER	String	No	Specifies the clock source for the first register on the chainout_saturate input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*
CHAINOUT_SATURATE_ACLR	String	No	Specifies the asynchronous clear source for the first register on the chainout_saturate input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and CHAINOUT_SATURATE_REGISTER is used, the default value is ACLR3.*(1)*
CHAINOUT_SATURATE_PIPELINE_REGISTER	String	No	Specifies the clock source for the second register on the chainout_saturate input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*

Table 3–3. altmult_add Megafunction Parameters (Part 12 of 18)

Parameter	Type	Required	Comments
CHAINOUT_SATURATE_OUTPUT_REGISTER	String	No	Specifies the clock source for the third register on the chainout_saturate input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*
CHAINOUT_SATURATE_OUTPUT_ACLR	String	No	Specifies the asynchronous clear source for the third register on the chainout_saturate input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and CHAINOUT_SATURATE_OUTPUT_REGISTER is used, the default value is ACLR3. *(1)*
ACCUMULATOR	String	No	Specifies the accumulator mode of the final adder stage. Values are YES and NO. If omitted, the default value is NO. When value is set to YES, rounding is dynamic and you must initialize the accumulator while rounded data is acquired. *(3)*
CHAINOUT_ADDER	String	No	Specifies the chainout mode of the final adder stage. Values are YES and NO. If omitted, the default value is NO. *(3)*
ZERO_CHAINOUT_OUTPUT_REGISTER	String	No	Specifies the clock source for the first register on the zero_chainout input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*
ZERO_CHAINOUT_OUTPUT_ACLR	String	No	Specifies the asynchronous clear source for the first register on the zero_chainout input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and ZERO_CHAINOUT_OUTPUT_REGISTER is used, the default value is ACLR3. *(1)*
ZERO_LOOPBACK_REGISTER	String	No	Specifies the clock source for the first register on the zero_loopback input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*

Table 3–3. altmult_add Megafunction Parameters (Part 13 of 18)

Parameter	Type	Required	Comments
ZERO_LOOPBACK_ACLR	String	No	Specifies the asynchronous clear source for the first register on the zero_loopback input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and ZERO_LOOPBACK_PIPELINE_REGISTER is used, the default value is ACLR3. *(1)*
ZERO_LOOPBACK_PIPELINE_REGISTER	String	No	Specifies the clock source for the second register on the zero_loopback input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*
ZERO_LOOPBACK_PIPELINE_ACLR	String	No	Specifies the asynchronous clear source for the second register on the zero_loopback input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and ZERO_LOOPBACK_PIPELINE_REGISTER is used, the default value is ACLR3. *(1)*
ZERO_LOOPBACK_OUTPUT_REGISTER	String	No	Specifies the clock source for the third register on the zero_loopback input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*
ZERO_LOOPBACK_OUTPUT_ACLR	String	No	Specifies the asynchronous clear source for the third register on the zero_loopback input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and ZERO_LOOPBACK_OUTPUT_REGISTER is used, the default value is ACLR3.*(1)*
ACCUM_SLOAD_REGISTER	String	No	Specifies the clock source for the first register on the accum_sload input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*
ACCUM_SLOAD_ACLR	String	No	Specifies the asynchronous clear source for the first register on the accum_sload input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and ACCUM_SLOAD_REGISTER is used, the default value is ACLR3. *(1)*

Table 3–3. altmult_add Megafunction Parameters (Part 14 of 18)

Parameter	Type	Required	Comments
ACCUM_SLOAD_PIPELINE_REGISTER	String	No	Specifies the clock source for the second register on the accum_sload input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*
ACCUM_SLOAD_PIPELINE_ACLR	String	No	Specifies the asynchronous clear source for the second register on the accum_sload input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and ACCUM_SLOAD_PIPELINE_REGISTER is used, the default value is ACLR3. *(1)*
SHIFT_MODE	String	No	Specifies the shift mode. Values are NO, LEFT, RIGHT, ROTATION, and VARIABLE. If VARIABLE is selected, rotate and shift_right are used to specify shift left, shift right, or rotation. If omitted, the default value is NO. *(1)* **Note**: This parameter is supported only when inputs equal 32 bits each, output equals 32 bits, and the number of multipliers equals 1.
ROTATE_REGISTER	String	No	Specifies the clock source for the first register on the rotate input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*
ROTATE_ACLR	String	No	Specifies the asynchronous clear source for the first register on the rotate input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and ROTATE_REGISTER is used, the default value is ACLR3. *(1)*
ROTATE_PIPELINE_REGISTER	String	No	Specifies the clock source for the second register on the rotate input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*
ROTATE_PIPELINE_ACLR	String	No	Specifies the asynchronous clear source for the second register on the rotate input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and ROTATE_PIPELINE_REGISTER is used, the default value is ACLR3. *(1)*

Table 3–3. altmult_add Megafunction Parameters (Part 15 of 18)

Parameter	Type	Required	Comments
ROTATE_OUTPUT_REGISTER	String	No	Specifies the clock source for the third register on the rotate input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*
ROTATE_OUTPUT_ACLR	String	No	Specifies the asynchronous clear source for the third register on the rotate input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and ROTATE_OUTPUT_REGISTER is used, the default value is ACLR3. *(1)*
SHIFT_RIGHT_REGISTER	String	No	Specifies the clock source for the first register on the shift_right input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*
SHIFT_RIGHT_ACLR	String	No	Specifies the asynchronous clear source for the first register on the shift_right input. Values are NONE, ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and SHIFT_RIGHT_REGISTER is used, the default value is ACLR3. *(1)*
SHIFT_RIGHT_PIPELINE_REGISTER	String	No	Specifies the clock source for the second register on the shift_right input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*
SHIFT_RIGHT_PIPELINE_ACLR	String	No	Specifies the asynchronous clear source for the second register on the shift_right input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and SHIFT_RIGHT_PIPELINE_REGISTER is used, the default value is ACLR3.*(1)*
SHIFT_RIGHT_OUTPUT_REGISTER	String	No	Specifies the clock source for the third register on the shift_right input. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(1)*

Table 3–3. altmult_add Megafunction Parameters (Part 16 of 18)

Parameter	Type	Required	Comments
SHIFT_RIGHT_OUTPUT_ACLR	String	No	Specifies the asynchronous clear source for the third register on the shift_right input. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and SHIFT_RIGHT_OUTPUT_REGISTER is used, the default value is ACLR3. *(1)*
PORT_OUTPUT_IS_OVERFLOW	String	No	Specifies port usage. Values are PORT_UNUSED and PORT_USED. When the value is set to PORT_USED, output pin overflow is added. If omitted, the default value is PORT_UNUSED. *(1)*
PORT_CHAINOUT_SAT_IS_OVERFLOW	String	No	Specifies port usage. Values are PORT_UNUSED and PORT_USED. When the value is set to PORT_USED, output pin chainout_sat_overflow is added. If omitted, the default value is PORT_UNUSED. *(1)*
EXTRA_LATENCY	String	No	Specifies the number of clock cycles of latency.
LPM_HINT	String	No	Allows you to specify Altera-specific parameters in VHDL Design Files (**.vhd**). The default is UNUSED.
LPM_TYPE	String	No	Identifies the library of parameterized modules (LPM) entity name in VHDL design files.
INTENDED_DEVICE_FAMILY	String	No	This parameter is used for modeling and behavioral simulation purposes. Create the altmult_add megafunction with the MegaWizard Plug-in Manager to calculate the value for this parameter.
DSP_BLOCK_BALANCING	String	No	If omitted, the default is AUTO.
ADDNSUB[]_ROUND_ACLR	String	No	Parameter [1,3]. Specifies the asynchronous clear source for the first register on the corresponding addnsub[]_round input port. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and corresponding ADDNSUB[]_ROUND_REGISTER is used, the default is ACLR3. *(2)*

Table 3–3. altmult_add Megafunction Parameters (Part 17 of 18)

Parameter	Type	Required	Comments
ADDNSUB[]_ROUND_ PIPELINE_ACLR	String	No	Parameter [1,3]. Specifies the asynchronous clear source for the second register on the corresponding addnsub[]_round input port. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and corresponding ADDNSUB_[]_ ROUND_PIPELINE_REGISTER is used, the default is ACLR3. *(2)*
ADDNSUB[]_ROUND_ PIPELINE_REGISTER	String	No	Parameter [1, 3]. Specifies the clock source for the second register on the corresponding addnsub[]_round input port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(2)*
ADDNSUB[]_ROUND_ REGISTER	String	No	Parameter [1,3]. Specifies the clock source for the first register on the corresponding addnsub[]_round input port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(2)*
DEDICATED_MULTIPLIER_ CIRCUITRY	String	No	Specifies whether to use the DSP block to implement the circuit. Values are YES, NO, and AUTO. The circuit is implemented using the DSP block when the value is set to YES If omitted, the default is AUTO.
MULT[]_ROUND_ACLR	String	No	Parameter [01,23]. Specifies the asynchronous clear source for the second register on the corresponding mult[]_round input port. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and corresponding MULT[]_ROUND_REGISTER is used, the default is ACLR3. *(2)*
MULT[]_ROUND_REGISTER	String	No	Parameter [01,23]. Specifies the clock source for the register on the corresponding mult[]_round input port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(2)*

Table 3–3. altmult_add Megafunction Parameters (Part 18 of 18)

Parameter	Type	Required	Comments
MULT[]_SATURATION_ACLR	String	No	Parameter [01,23]. Specifies the asynchronous clear source for the register on the corresponding mult[]_saturation input port. Values are ACLR0, ACLR1, ACLR2, and ACLR3. If omitted and corresponding MULT[]_SATURATION_REGISTER is used, the default is ACLR3. *(2)*
MULT[]_SATURATION_REGISTER	String	No	Parameter [01,23]. Specifies the clock source for the register on the corresponding mult[]_saturation input port. Values are UNREGISTERED, CLOCK0, CLOCK1, CLOCK2, and CLOCK3. If omitted, the default value is CLOCK0. *(2)*

Note to Table 3–3:
(1) This parameter is available only for Stratix III devices.
(2) This parameter is available only for Stratix II devices.
(3) This parameter is available only for Stratix II and Stratix III devices.